НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

(повне найменування інституту, факультету)

Автоматизованих систем обробки інформації і управління

(повна назва кафедри)

«До захисту допущено»
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис) (ініціали, прізвище)
“ ” 2020 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Програмне забезпечення інформаційних управляючих систем та технологій»

спеціальності «121 Інженерія програмного забезпечення»

на тему *Веб-застосування для аналізу рівня глюкози у крові діабетиків лінгвістичним методом*

Виконав: студент IV курсу, групи *ІП-61 Шаверський Іван Олександрович* _____
(прізвище, ім’я, по батькові) (підпис)

Керівник *асистент Очеретяний О.К.* _____
посада,науковий ступінь,вчене звання,прізвище,і ім’я, по батькові (підпис)

Консультант з графічної документації *доцент, к.т.н., Ліщук К.І.* _____
посада,науковий ступінь,вчене звання,прізвище,і ім’я, по батькові (підпис)

Рецензент:

доц., к.т.н., Кисленко Юрій Іванович
посада,науковий ступінь,вчене звання,прізвище,ім’я, по батькові (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

UNICHECK

Власник документу:
Попенко Володимир Дмитрович

Дата перевірки:
14.06.2020 01:09:23 EEST

Дата звіту:
14.06.2020 18:18:41 EEST

ID перевірки:
1004022419

Тип перевірки:
Doc vs Internet + Library

ID користувача:
77149

Назва документу: Shaverskij_ip61

ID файлу: 1004035490 **Кількість сторінок:** 60 **Кількість слів:** 10701 **Кількість символів:** 77828 **Розмір файлу:** 274.43 KB

6% Схожість

Найбільша схожість: 2.19% з джерело бібліотеки. **ID файлу:** 1004035487

2.88% Схожість з Інтернет джерелами	29	Page 62
5.65% Текстові збіги по Бібліотеці акаунту	160	Page 62

0.43% Цитат

Цитати	1	Page 63

Вилучення переліку посилань вимкнено

0% Вилучень

Вилучений текст відсутній

Підміна символів

Заміна символів	18

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет (інститут) *Інформатики та обчислювальної техніки*
(повна назва)

Кафедра *автоматизованих систем обробки інформації і управління*
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – *121 Інженерія програмного забезпечення*

Освітньо-професійна програма – *Програмне забезпечення інформаційних управляючих систем та технологій*

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис)
“___”_____2020 р.

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ СТУДЕНТУ

Шаверському Івану Олександровичу
(прізвище, ім’я, по батькові)

1. Тема проєкту « *Веб-застосування для аналізу рівня глюкози у крові діабетиків лінгвістичним методом* »
керівник проєкту *Очеретяний Олександр Констянтинович, ассистент*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від “07” травня 2020 р. №1081-с

2. Термін подання студентом проєкту «08» *червня 2020 року*

3. Вихідні дані до проєкту
Технічне завдання

4. Зміст пояснювальної записки
1) Аналіз вимог до програмного забезпечення: основні визначення та терміни, опис предметного середовища, огляд існуючих технічних рішень та відомих програмних продуктів, розробка функціональних та нефункціональних вимог
2) Моделювання та конструювання програмного забезпечення: моделювання та аналіз програмного забезпечення, засоби розробки, технічні рішення, архітектура програмного забезпечення, аналіз безпеки даних
3) Методика тестування
4) Керівництво користувача, методика випробувань програмного продукту

5. Перелік графічного матеріалу

1) Схема структурна варіантів використань

2) Схема бази даних

3) Схема структурна класів програмного забезпечення

4) Креслення вигляду екранних форм

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «10» *березня 2020 року*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	*Вивчення рекомендованої літератури*	*28.02.2020*	
2.	*Аналіз існуючих методів розв'язання задачі*	*07.03.2020*	
3.	*Постановка та формалізація задачі*	*15.03.2020*	
4.	*Аналіз вимог до програмного забезпечення*	*01.04.2020*	
5.	*Алгоритмізація задачі*	*08.04.2020*	
6.	*Моделювання програмного забезпечення*	*15.04.2020*	
7.	*Обґрунтування використовуваних технічних засобів*	*21.04.2020*	
8.	*Розробка архітектури програмного забезпечення*	*01.05.2020*	
9.	*Розробка програмного забезпечення*	*02.05.2020*	
10.	*Налагодження програми*	*13.05.2020*	
11.	*Виконання графічних документів*	*18.05.2020*	
12.	*Оформлення пояснювальної записки*	*20.05.2020*	
13.	*Подання ДП на попередній захист*	*28.05.2020*	
14.	*Подання ДП рецензенту*	*05.06.2020*	
15.	*Подання ДП на основний захист*	*08.06.2020*	

Студент _____ Іван ШАВЕРСЬКИЙ
(підпис)

Керівник _____ Олександр ОЧЕРЕТЯНИЙ
(підпис)

№з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	КПІ.ІП-6126.045440.01.81	Пояснювальна записка	69	
3	А4	КПІ.ІП-6126.045440.02.91	Технічне завдання	11	
4	А4	КПІ.ІП-6126.045440.03.13	Опис програми	34	
5	А4	КПІ.ІП-6126.045440.04.51	Програма та методика тестування	6	
6	А3	КПІ.ІП-6126.045440.05.34	Керівництво користувача	15	
7	А3	КПІ.ІП-6126.045440.06.99.СС	Схема структурна варіантів використання	1	
8	А3	КПІ.ІП-6126.045440.06.99.СБД	Схема бази даних	1	
9	А3	КПІ.ІП-6126.045440.06.99 КЕ	Креслення вигляду екранних форм	1	
10	А3	КПІ.ІП-6126.045440.06.99.СС	Схема структурна класів програмного забезпечення	2	

Зм.	Арк.	ПІБ	Підп.	Дата	КПІ.ІП-6126.045440.00.90		
Розробн.		Шаверський І.О.			Відомість дипломного проєкту	Літ.	Лист 1 / Листів 1
Керівн.		Очеретяний О.К.					
Консульт						КПІ ім.Ігоря Сікорського кафедра АСОІУ гр. ІП-61	
Н/контр.		Ліщук К.І.					
В.о.зав.каф.		Павлов О.А.					

АНОТАЦІЯ

Пояснювальна записка складається із 4 розділів та містить 15 рисунків, 21 таблицю, 5 додатків та 39 джерел – загалом 155 сторінок.

Метою проекту є розробка веб-застосування, яке б допомагало користувачам-діабетикам з контролем їх рівня глюкози надаючи можливість записувати свій рівень глюкози та на його основі отримувати графіки та аналітику. Аналітична складова виконана з використанням лінгвістичного методу.

У розділі «Аналіз вимог до програмного забезпечення» проаналізована предметна область та вимоги до програмного забезпечення. Також були розглянуті існуючі технічні рішення та успішні програмні продукти, розроблені функціональні та нефункціональні вимоги.

У розділі «Моделювання та конструювання програмного забезпечення» було розроблено архітектуру платформи, наведена діаграма класів та використані інструменти. В даному розділі описано такі складові серверного програмного забезпечення, як авторизація, зберігання даних та опис бізнеслогіки сервісу.

У розділі «Аналіз якості та тестування програмного забезпечення» наведений розроблений план тестування та результати його проведення.

У розділі «Впровадження та супровід програмного забезпечення» описано процес розгортання сервера та інструкція користувача.

Ключові слова: ДІАБЕТ, ГЛЮКОЗА, ВЕБ ЗАСТОСУНОК, СЕРВЕРНІ СИСТЕМИ, ЛІНГВІСТИЧНИЙ МЕТОД, ГІПЕРГЛІКЕМІЯ, ГІПОГЛІКЕМІЯ

ABSTRACT

The explanatory note consists of 4 sections and contains 15 figures, 21 tables, 5 appendices and 39 sources - a total of 155 pages.

The aim of the project is to develop a web application that would help diabetic users to control their glucose levels by enabling them to record their glucose levels and get charts and analytics based on them. The analytical component is performed using the linguistic method.

In the section "Analysis of software requirements" the subject area and software requirements are analyzed. Existing technical solutions and successful software products were also considered, functional and non-functional requirements were developed.

In the section "Software modeling and design" the platform architecture was developed, used tools and class diagram is given. This section describes such components of the server software as authorization, data storage and service business logic description.

The section "Quality analysis and software testing" shows the developed testing plan and the results of its implementation.

The "Software Deployment and Maintenance" section describes the server deployment process and user instructions.

Key words: DIABETES, GLUCOSE, WEB APPLICATION, SERVER SYSTEMS, LINGUISTIC METHOD, HYPERGLYCAEMIA, HYPOGLYCAEMIA

Пояснювальна записка
до дипломного проєкту

на тему: Веб-застосування для аналізу рівня глюкози у крові діабетиків _____
лінгвістичним методом_____
_____

Київ – 2020 року

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

1) англ. – англійською.
2) мг/дл – міліграми на децилітр.
3) ммоль/л – мілімоль на літр.
4) Скрипт – програма, яка має на меті автоматизацію деякої задачі.
5) Бібліотека – зібрання деяких підпрограм, що використовуються для рішення певних схожих типових задач та застосовуються для пришвидшення та полегшення розробки програмного забезпечення.
6) HTML – мова розмітки гіпертексту. Використовується для розробки веб-сайтів.
7) Веб-сайт – деяка сукупність веб-сторінок, які доступні у мережі Інтернет, об’єднані за змістом і доменним ім’ям.
8) Браузер – програмне забезпечення, що використовується у глобальній мережі для запиту, обробки, маніпулювання та відображення веб-сайтів.
9) Машинне навчання – вивчення алгоритмів та математичних моделей, які комп’ютерні системи використовують для ітеративного покращення їх продуктивності для конкретної задачі.
10) Веб-дизайн – процес створення вигляду веб-сайтів.
11) ПЗ – Програмне забезпечення.
12) UNIX – родина багатозадачних та багатокористувацьких операційних систем, що походять від AT&T Unix, розробка якої почалася в 1970-х в дослідницькому центрі Bell Labs grep Кеном Томпсоном, Деннісом Рітчі та іншими.
13) Реляційна форма (реляційна модель) – підхід в управлінні даними, де усі дані представлені за у вигляді відношень (таблиць).
14) API – набір процедур та функцій, які дозволяють створювати додатки з взаємодією різних компонентів, наприклад, доступом до даних операційної системи, додатку, чи іншого сервісу.

15) Сигмоїда — це неперервно диференційована нелінійна монотонна S-подібна функція, яка часто застосовується для «згладжування» значень деякої величини. Вона часто використовується в нейронних мережах для такого, аби зробити роботу мережі більш нелінійною, не сильно впливаючи на результат роботи.
16) Фітнес трекер – зовнішній пристрій або мобільний застосунок, який займається моніторингом показників, що пов’язані з фітнесом пройдена відстань, споживання калорій, показники серцевого ритму та якості сну. Є одним з видів носимого комп'ютера.
17) JSON – текстовий формат обміну даних.

ВСТУП

Відповідно до даних Міжнародної діабетичної федерації, близько 425 мільйонів людей по всьому світу хворіють цукровим діабетом (при цьому половина з них не знає про це). Цукровий діабет – це хронічна хвороба основним симптомом якої є підвищення кількості цукру в крові. Основним фактором ризику при даній хворобі є виникнення гіпоглікемії та гіперглікемії. Гіпоглікемія це стан при якому рівень глюкози опускається нижче 70 мг/дл (3.9 ммоль/л). Гіперглікемія це стан при якому рівень глюкози тримається вище ніж 200 мг/дл (11.11 ммоль/л) через 7-8 годин після останнього прийому їжі.

Найбільш розповсюдженими наслідками гіпоглікемії є різноманітні проблеми з вестибулярним апаратом (порушення координації рухів, нудота), мозковою активністю (амнезія, примітивні автоматизми, епілептичні розлади, втрата свідомості розлади дихання, парастензії тощо), проблеми з сердцем (аритмія). З наслідків гіперглікемії можна виділити серцево-судинні захворювання, невропатія, проблеми з кістками та суглобами. Проте найбільш критичні та небезпечні для життя людини наслідки цих станів – діабетичний кетоацидоз, гіпоглікемічна кома та гіперглікемічна кома. В обох випадках якщо не буде надана своєчасна медична допомога даний критичний стан загрожує комою або навіть смертю особи. Таким чином своєчасне виявлення ризику через-мірного пониження або підвищення рівня глюкози може допомогти тисячам людей і навіть спасти їх життя.

Тож основною метою розробки є створення WEB-застосунку для збору статистики по рівню цукру в крові та передбаченню ризиків його пониження або підвищення.

Дана статистика також може бути корисною при вивченні цукрового діабету і хвороб пов’язаних з ним. (Статистика може бути зібрана лише з дозволу користувачів даного WEB-застосунку)

Відповідно до статистики ВООЗ (Всесві́тня організація охорони здоров'я) кількість хворих на діабет є дуже великою і постійно зростає тож дана тема і розроблений застосунок є і будуть залишатися актуальними.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

Було прийнято рішення виконати цю програму саме у вигляді веб-застосунку [1]. Веб-застосунок це розподілене програмне забезпечення у якому клієнтом є браузер, а сервер це веб-сервер. Такий вид застосунків набув популярності у кінці 1990-х – початку 2000-х років. Популярним його зробила надзвичайна гнучкість у оточенні з якого нею можна користуватись – веб-застосунки не залежать від операційної системи, а залежать від браузера (якщо точніше то від ядра, базисної частини браузера). Таким чином одним і тим же веб застосунком можна користуватися з Windows, Linux, MacOS, Android (з деякими особливостями розмітки) або будь якого іншого оточення на якому встановлений такий браузер, який буде підтримувати застосовані для «верстання» (розробки розмітки сайту, його візуального вигляду) або написання javascript (мультипарадигменна мова програмування яка здавна э основною мовою для написання клієнтного коду для браузера) коду функції. За такої організації програми можна вважати, що браузер є реалізацією тонкого клієнту. При використанні тонкого клієнту вся основна логіка застосунку зосереджена на сервері, а клієнт є лише інтерфейсом для користувача для взаємодії з сервером та відображає наявну на ньому інформацію. В нашому випадку використання веб-застосунку це можливість збільшити аудиторію, що надзвичайно важливо для ефективної роботи основного алгоритму та спростити розробку застосунку.

Ніякий аналіз даних не може обійтись без власне даних. Основними у цьому застосунку є введені користувачем дані про рівень глюкози з уточненням часу даного виміру. Як відомо цей показник постійно змінюється через роботу підшлункової залози і інсуліном, який вона виробляє. Саме основну динаміку зміни рівня глюкози програмний застосунок відслідковує і прогнозує.

При цьому для початку аналізу і прогнозування було знайдено загальнодоступний дата-сет D1NAMO [2] зібраний

у Швейцарії декількома дослідниками. Це найбільш повний на даний момент дата-сет з сирих, не агрегованих та не оброблених даних для досліджень пов'язаних з діабетом. Він включає в себе показники від 20 здорових осіб та 9 осіб, що мають діабет першого типу. Спостереження проводились протягом 6 днів (за деякими виключенням). За цей час були зібрані такі дані:

– постійні заміри рівня глюкози (приблизно раз в 5 хвилин) зібрані засобами автоматизованого заміру глюкози;

– записи про приймання інсуліну (для діабетиків);

– детальна електрокардіограма;

– показники дихання;

– дані акселерометра;

– фото або опис їжі.

На основі цих даних і буде проводитись загальний аналіз і передбачення.

З подібних дата-сетів для цілей даного проекту підходив ще тільки UCI Diabetes [3]. В його наповнені приймало участь набагато більше людей, але дані вимірів глюкози були меньш деталізованими, лише по 3-8 замірів на день.

Відповідно до збільшення кількості користувачів кількість даних для аналізу та передбачення буде збільшуватись, що в свою чергу покращить точність передбачуваного значення.

Тож основними задачами під час розробки цього продукту була реалізація алгоритму пошуку передбачуваного значення та розробка зручного інтерфейсу для залучення нових користувачів для збільшення кількості даних для подальших прогнозів.

В реальних умовах, на відміну від умов з яких проект розпочинається (доступні виміри глюкози кожні 5 хвилин) люди не завжди будуть заносити показники засобів автоматизованого заміру глюкози. Тобто у нас буде набагато менше даних, що негативно вплине на працездатність лінгвістичного методу. Щоб вирішити цю проблему буде використано інтерполяцію [4].

Інтерполяція – це поняття з обчислювальної математики. Це спосіб знаходження проміжних значень за деяким наявним набором значень відомих. Отже, за допомогою інтерполяції ми зможемо «збільшити» наш об'єм даних для того, щоб підтримати дієздатність веб-застосунку. Але спершу треба вибрати, яку саме інтерполяцію ми будемо використовувати.

Інтерполяція методом найближчого сусіда не підходить так як вихідна множина значень все ще не буде задовольняти потреб алгоритму. Лінійна інтерполяція вже набагато краще, так як таким чином ми можемо отримати потрібну кількість значень. Проте, хоч вона і дуже швидко та легко обчислюється, вона теж не підходить через недостатню «якість» значень (вони будуть не дужі схожі на реальні)

Є ще три загальновідомі та використовувані варіанти: Метод скінчених різниць, Інтерполяційний поліном Лагранжа та Сплайн інтерполяція. З них було обрано Сплайн інтерполяцію (А саме інтерполяцію кубічними сплайнами [5]) через достатню «якість» та відносну легкість та швидкість обчислення.

Для побудови кубічного сплайна необхідно побудувати n многочленів третьої степені:

$$S(x) = a_i + b_i(x - x_{i-1}) + c_i(x - x_{i-1})^2 + d_i(x - x_{i-1})^3, x_{i-1} \le x \le x_i, i = 1,2, \dots, n \quad 1.1$$

З достатньою кількістю даних можна переходити до власне аналізу цих даних, що і є основою цього проекту.

1.2 Змістовний опис і аналіз предметної області

Основою даної роботи є лінгвістичний метод, а саме лінгвістичне моделювання [6][7]. Воно базується на трьох основних підходах: інтервальні обчислення та робастні методи, математична лінгвістика та структурний підхід і сучасні методи ймовірнісного моделювання. Даний метод часто використовується в біоінформатиці для порівняння генів , хромосом та білків.

Лінгвістична модель це сукупність лінгвістичних (символьних) послідовностей, що були знайдені за допомогою лінгвістичного моделювання використовуючи деякі параметри лінгвістизації, а також формальна граматика, що була відновлена на основі цієї сукупності.

Саме ж лінгвістичне моделювання розглядається, як специфічний вид математичного моделювання, який використовується для обробки даних не у чисельному форматі, а в символьному. Для переходу з деякої множини числових значень до обраного алфавіту потрібно розбити цю множину на чисельні інтервали.

Нехай X та Y - дві частково впорядкованих множини. Ми позначимо, як B(X) та B(Y) множини усіх підмножин множин X та Y.

Впорядкованою називається множина, на якій задано відношення порядку, за умови, що відношення порядку визначено для будь-яких двох його елементів. Частково впорядкована – у протилежному випадку.

Структурою називається та частково впорядкована множина у якої будь яка її двоелементна підмножина має точну нижню та верхню межу. А повною структурою частково впорядкована множина називається у випадку, коли кожна її непуста підмножина має такі точні межі.

Кожну підмножину з множин X та Y будемо вважати умовно-повними структурами, а також позначимо, як S(X) та S(Y). Відношення порядку будемо позначати через ≤.

Якщо $a, b \in S(X)$ та a ≤ b, то множину $l(a,b) = [a,b] = \{x \in X, a \leq x \leq b\}$, будемо називати інтервалом на S(X).

Через $\zeta_{S(X)}$ будемо позначати множину усіх інтервалів на структурі S(X).За цих позначень, якщо $X = R^1$ - множина дійсних чисел, то ζ_{R^1} це множина закритих інтервалів на прямій дійсних чисел. В даному випадку її ще називають інтервальних числом.

Для даної системи інтервали мають бути рівночастотні. Тобто при рівнозначній інтеврвалізації N-того рівня множини X:

$$\omega(a_1, b_1) = \omega(a_2, b_2) = \omega(a_3, b_3) = \cdots = \omega(a_N, b_N) \quad 1.2$$

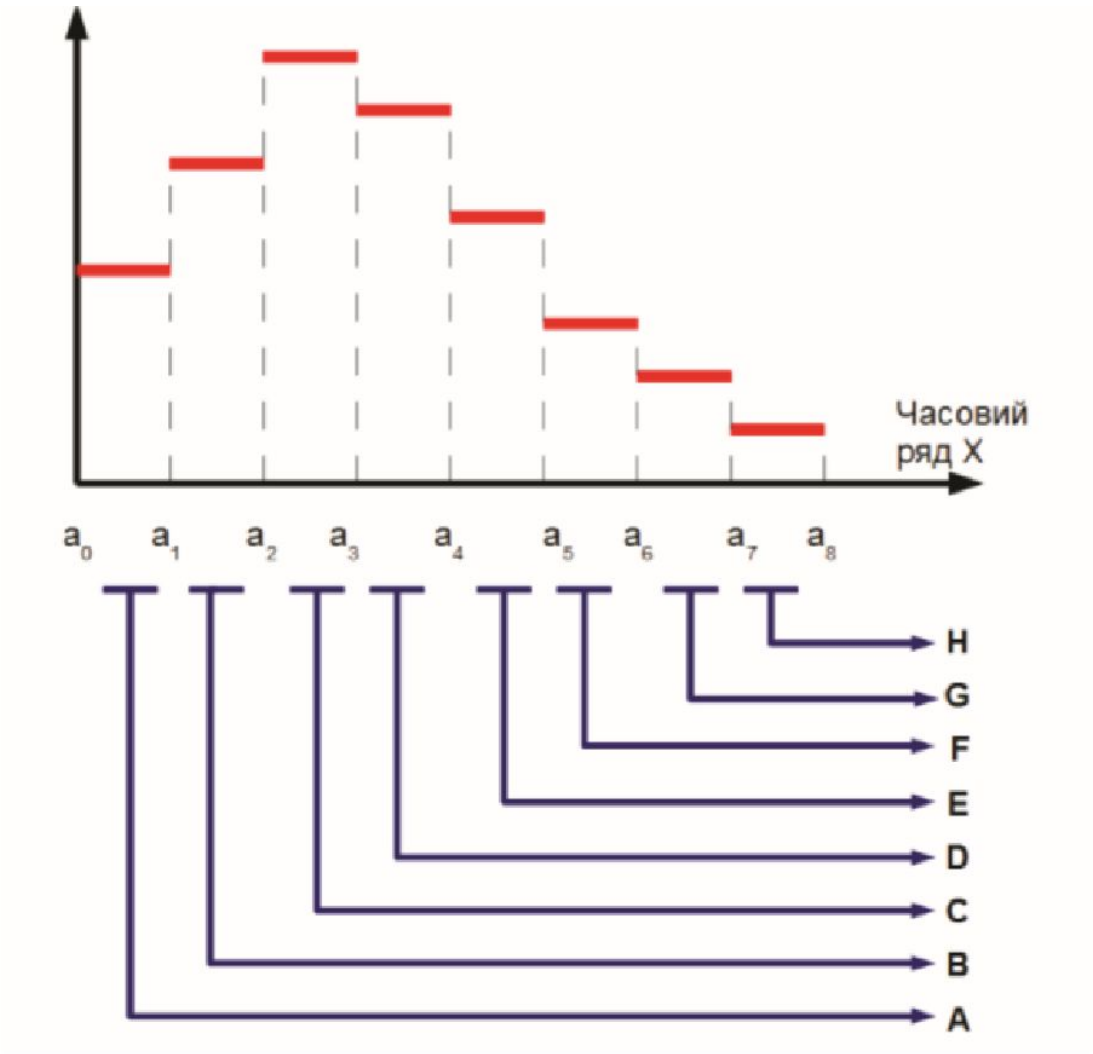


Рисунок 1.1 – Схема перетворення чисельних даних до символьного вигляду

На рисунку 1.1 представлена умовна схема лінгвістизації при рівнозначній інтервалізації. Інтервалізація дає множину інтервалів $l[a_0, a_1]$, $l[a_1, a_2], \ldots, l[a_7, a_8]$. За віссю абсцис розташовані впорядковані значення часового ряду та інтервали. За віссю ординат ранджується кількість значень часового ряду, що потрапили до відповідного інтервалу. Кількість елементів, які потрапляють до певного інтервалу (значення часового ряду можуть повторюватися) будемо позначати через $D\{I[a_i, a_{i+1}]\}$. Кількість інтервалів - M < N. Легко бачити, що сума $\sum_{i=0}^{M-1} D\{I[a_i, a_{i+1}]\} = N$.

Нарешті, ми можемо ввести поняття частотності інтервалу $I[a_i, a_{i+1}]$ на часовому ряді потужністю N:

$$\omega\upsilon_{i,i+1} = \upsilon[a_i, a_{i+1}] = \frac{D\{I[a_i, a_{i+1}]\}}{N} \quad 1.3$$

При цьому очевидно, що $\sum_{i=0}^{M-1} \upsilon_{i,i+1} = 1$, а враховуючи що для усіх і $\upsilon_{i,i+1} \geq \varnothing$ будемо мати аналогію аксіоматики теорії ймовірності.

Рівень глюкози у це дуже змінна величина, яка по різному коливається у здорових людей та у людей з розладами та порушенням толерантності до глюкози. Діабет небезпечний, якраз своїми непередбачуваними на перший погляд скачками рівня глюкози. Використання лінгвістичного методу для обробки значень рівня глюкози теж буде складатись с обробки числового ряду та перетворення його на лінгвістичну послідовність. Досі не має якось засобу, який би міг замінити професійного лікаря у допомозі людині хворій на діабет при підтриманні рівня глюкози на відносно здоровому рівні. Усі вони чимось не можуть повністю задовольнити потреби цієї проблеми.

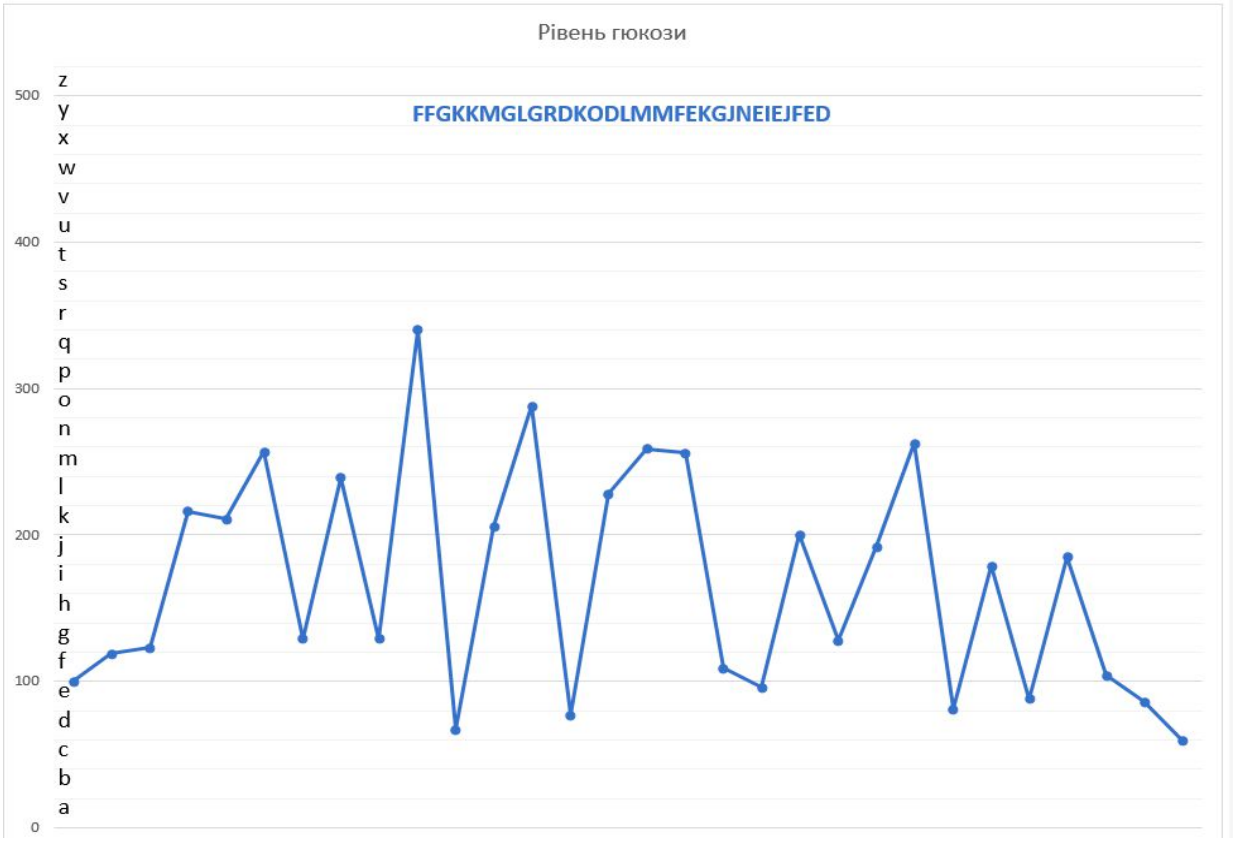


Рисунок 1.2 – Рівень глюкози перетворений на лінгвістичну послідовність

Наприклад, перетворимо ряд значень глюкози (зазначений у мг/дл) «100, 119, 123, 216, 211, 257, 129, 239, 129, 340, 67, 206, 288, 77, 228, 259, 256, 109, 96,

200, 128 ,192, 263, 81, 179, 88, 185, 104, 86, 60» на лінгвістичну послідовність. Використаємо англійський алфавіт (26 латинських символів) та рівномірний розподіл. З цими вхідними даними ми отримаємо лінгвістичний ряд «FFGKKMGLGRDKODLMMFEKGJNEIEJFED» (рис 1.2).

Одним з важливих питань для роботи з лінгвістичним методом є вибір алфавіту. Від нього частково залежить можливий розмір інтервалів та прямо залежить кількість цих інтервалів. Чим менше символів у алфавіті, тим більше в середньому значень буде потрапляти у інтервал відповідний до певного символу і навпаки – чим більше тим більш «точними» будуть наші інтервали, тобто з збільшенням алфавіту ми наближуємось, до обробки даних все більше схожих на вхідні, і ефективність методу після деякого часу почне зменшуватись.

Хорошими прикладами алфавітів може слугувати англійський алфавіт (це 26 латинських символів) або 100 послідовних символів таблиці ASCII (American standard code for information interchange) [8]. Перевагами англійського алфавіту є дуже наглядне відображення лінгвістичних послідовностей та розмір, що добре підходить під задачі з відносно невеликою потенційною множиною значень. Перевагами ж 100 послідовних символів таблиці ASCII очевидно є розмір, який буде корисним для задач с більшою цільовою множиною значень, але з недоліків варто виділити, що деякі символи з будь-яких 100 послідовних символів, які ми б не вибрали, не будуть наочно відображеними. Їх порядковий номер (та призначення) будуть відрізнятись, але розгледіти неозброєним оком різницю буде неможливо, прийдеться використовувати додаткові ідентифікатори при потребі оглянути візуально лінгвістичні ланцюги. Так, як потенційна множина значень рівня глюкози, навіть у людини с діабетом 1-го типу, відносно не велика, то для цього проекту було обрано саме англійську абетку, через достатній розмір та наочність.

Іншим дуже важливим питанням при використанні лінгвістичного методу є обрана функція розподілу ймовірностей [9]. Від неї буде залежати розмір кожного конкретного інтервалу (множена значень, що відповідає певному

символу) відносно інших інтервалів. Підбір розподілу може грати ключову роль у ефективності роботи лінгвістичного методу так як за допомогою нього ми можемо «концентруватись» на значеннях, які нас цікавлять і ще більше узагальнювати значення, різниця у коливанні, яких нас не цікавить.

Найпростішим вибором є рівномірний розподіл [10]. При використанні цієї функції розподілу розмір множини значень відповідної до кожного символу (розмір інтервалу) буде однаковий. Тобто ймовірність припадання значення до деякого інтервалу k_j дорівнює 1/n. Насправді, це непоганий вибір для багатьох задач, через швидкість роботи на ясність розподілу по інтервалам. Наприклад, непогано використовувати цей розподіл при аналізі ЕКГ (Електрокардіограм) лінгвістичним методом.

Але для задачі цієї роботи, а саме аналізу рівня глюкози у людей хворих на діабет, ми можемо підшукати розподіл, який буде краще збігатись з нашою потенційною множиною значень. У здорової людини нормальний рівень глюкози тримається на рівні 80-110 мг/дл коли вісім або більше годин людина нічого не їла, або 80-140 мг/дл протягом 2 годин після їжі. Поріг гіпоглікемії 70 мг/дл, тобто все, що менше уже надзвичайно небезпечно для людини. Гіперглікемію прийнято вважати 200 мг/дл та більше.

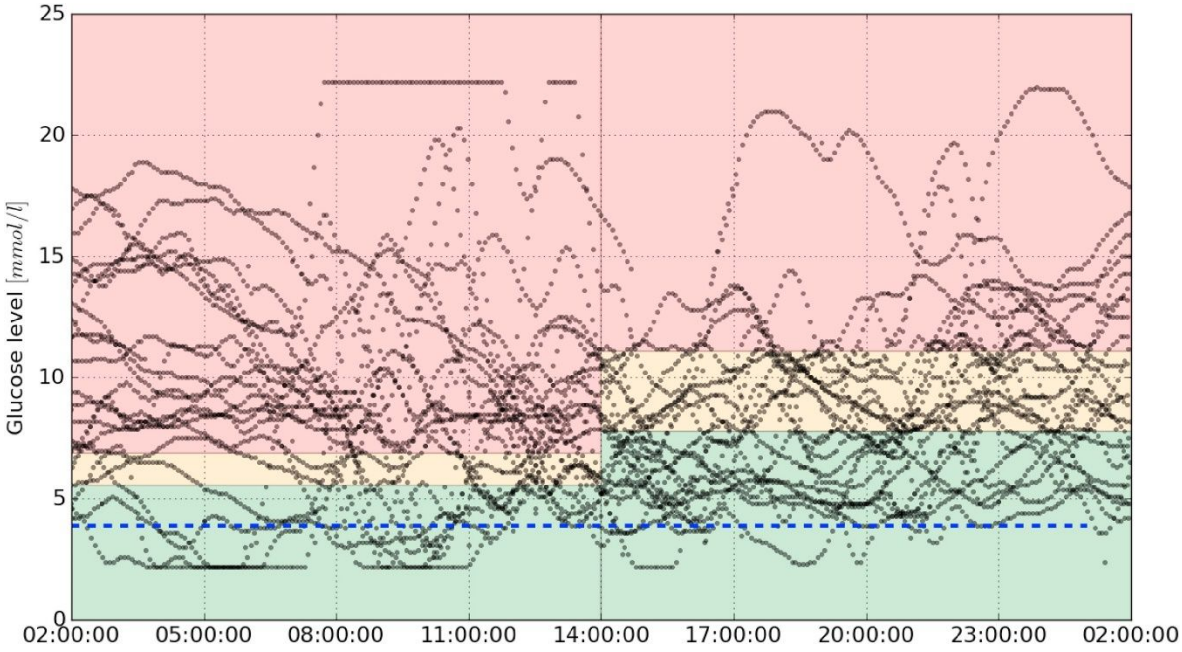


Рисунок 1.3 – Рівень глюкози у діабетиків дата-сету D1NAMO

Дослідивши обраним для проекту дата-сет D1NAMO [2], максимальне зафіксоване значення це 430 мг/дц (Рис 1.3). Виходячи з цього було вирішено встановити максимальне значення рівня глюкози в 520 мг/дц. Таким чином множина можливих значень рівня глюкози у даному веб-застосунку буде від 0 мг/дц до 520 мг/дц. З даних наведених вище ми знаємо, що значення глюкози від 70 до 200 мг/дл це значення не критичні отже їх можна об'єднувати в інтервали побільше.

Враховуючи це першим на думку спадає нормальний розподіл [11] (Також відомий, як розподіл Гауса чи Гауса-Лапласа). Формула цього розподілу:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \qquad 1.4$$

Тут σ це коефіцієнт розмірності, а μ коефіцієнт зсуву. На перший погляд цей розподіл нам підходить: маємо пік приблизно в тій зоні, де у нас здорові значення глюкози. Але через те, що розмір гіпоглікемічного вікна у нас набагато менше за вікно гіперглікемічне, ми будем «втрачати» дані, що відповідають стану гіперглікемії і сильно зменшимо ефективність прогнозування для цих значень.

Рисунок 1.4 – Графіки щільності логнормального розподілу з різними параметрами

Тут у нагоді нам стане логнормальний розподіл [12]. Це ще одне двопараметричне сімейство абсолютно неперервних розподілів. Основна їх ідея цього розподілу це, що логарифм випадкової величини має нормальний розподіл. Формула цього розподілу:

$$f(x) = \frac{1}{x\sqrt{2\pi\sigma^2}} e^{-\frac{(\ln x - \mu)^2}{2\sigma^2}} \quad 1.5$$

Тут σ це коефіцієнт розмірності, а μ коефіцієнт зсуву. Як можемо бачити з графіку щільності логнормального розподілу (Рис 1.4) спад функції є більш плавним, а тому і більше підходить під нашу предметну область.

Останнім, але не менш важливим для цієї роботи є побудування матриці появи деякого символу після певної послідовності символів, а також на її основі та даних про узагалі частоту появи певного символу побудування матриці шансу появи деякого символу після певної послідовності символів. Вона необхідна нам, щоб безпосередньо робити якісь передбачення, тобто дізнаватися, який символ з найбільшою ймовірність буде після, якоїсь послідовності символів.

Наприклад візьмемо алфавіт, що складається з 5 символів A, B, C, D та E і спробуємо створити побудувати матрицю появ символів один за одним для лінгвістичної послідовності ABBCEDDDDEAC. Вигляд цієї матриці можна побачити на рисунку 1.5.

	A	B	C	D	E
A	0	1	1	0	0
B	0	1	1	0	0
C	0	0	0	0	1
D	0	0	0	3	1
E	1	0	0	1	0

Рисунок 1.5 – Матрицю появ символів один за одним лінгвістичної послідовності ABBCEDDDDEAC

І тепер маючи цю матрицю та дані про частоту появ символів (підрахунок кількості входжень певного символу у лінгвістичній послідовності) можемо дізнатись шанс появи якогось символу після іншого. Наприклад, значи, що В зустрічалась у цій послідовності усього 2 рази, то шанс появи символу В після символу А рівний 50%.

Тепер коли у нас є все необхідне для коректної роботи лінгвістичного метода ми можемо сказати, який приблизно алгоритм роботи нашого веб-застосунку для попередження гіпоглікемічних та гіперглікемічних станів: спочатку зчитуємо дані (з файлу або від користувача). Якщо нас не влаштовує частота замірів ми інтерполюємо для отримання більшої кількості data points (замірів глюкози). Далі ми перетворюємо заміри глюкози за весь час для однієї людини на лінгвістичну послідовність використовуючи лінгвістичний метод з логнормальним розподілом та англійською абеткою. З отриманому наборі символів будуємо матрицю появ певного символу після деякої послідовності символів залежно від проміжку часу на який потрібно передбачити і використовуючи її та інформацію про кількість входжень всіх символів у лінгвістичну послідовність можемо знайти найймовірніший наступний символ послідовності. Далі повторюємо операцію поки не досягнемо постійної кількості кроків.

1.3 Аналіз успішних ІТ-проектів

1.3.1 Аналіз відомих технічних рішень

У даній роботі можна порівняти, як і види самих застосунків так і способу аналізу та обробки значень рівня глюкози. Подібні цьому застосунки зазвичай робляться у вигляді веб-застосунку, мобільного додатку або десктопного «товстого» клієнта (Коли весь застосунок не тільки є інтерфейсом до сервера, а й сам є сервером).

«Товстий» клієнт накладає обмеження на користувача: він може і повинен буде користуватись застосунком тільки на одному пристрої, а перенос даних на

інший засіб дуже часто або неможливий, або не зручний чи неавтоматизований. Тому цей варіант не є оптимальним для даного ПЗ.

Інший можлива реалізація цього проекту це мобільний додаток (на смартфони). Станом на сьогодні мобільний телефон це один із найпопулярніших, якщо не найпопулярніший технічний засіб, яким користується переважна більшість людей. Досить зручно було б записувати рівень своєї глюкози у будь-якому місці не витрачаючи багато часу на запуск комп'ютера або діставання ноутбука. На ринку наявні, як і прості додатки-записники для того, що просто записувати свій рівень глюкози на деякі інші дані на свій телефон, так і додатки, які пропонують аналітику та деякі інші додаткові функції.

Проте було вирішено зробити саме веб-застосунок. Так як веб-застосунок залежний лише від можливостей браузера, то можливо зробити його так, щоб ним була можливість зручно користуватись як і на комп'ютері так і на телефоні маючи при цьому всього один додаток, а не декілька зроблені спеціально під різні платформи.

Тепер, щодо способів аналізу та обробки значень рівня глюкози. Мною було знайдено 3 найчастіших рішення:

- нейронні мережі з довгою короткостроковою пам'яттю (LSTM);
- моделі з перемиканням режимів (Regime-switching AR models);
- реактивний підхід.

Нейронні мережі це на сьогодні один із найчастіше використовуваних способів аналізу даних з метою розпізнавання образів. В даному випадку нам потрібно розпізнавати потенційні підйоми та спади рівня глюкози. Хорошим прикладом такого застосування нейронних мереж є використання нейронних мереж довгої короткострокової пам'яті для декількох часових рядів досліджене у статті Predicting Blood Glucose Dynamics with Multi-time-series Deep Learning [13].

Там було використано MT-LSTM (Multi-Timescale Long Short-Term Memory Neural Networks) [14] на зібраному дослідниками дата-сеті у наповненні

якого приймали участь 112 людей хворих на діабет. Основна особливість такої нейронної мережі у тому, що вона здатна навчатися довгостроковим залежностям – на відміну від більшості нейронних мереж вони відразу сконструйовані так, що б запам’ятовувати інформацію на довгі періоди часу, а не тяжко навчатися цьому з нуля. В основі роботи цієї технології поняття «комірки», яка є якби одним з конвеєрів по якому мають пройти дані. Сама ж «комірка» складається с 4 сігмоідальних шарів та 3 фільтрів.

За результатами цього дослідження така мережа дуже добре справляється з передбаченням понижених (гіпоглікемічних) рівнів глюкози (Точність 95%). З нормальним та зависоким рівнем глюкози ситуація гірша. Всього 85 відсотків точність для нормального та 75-80 для зависокого. Також структура нейронної мережі потребує багато обчислень, отже є повільним для обрахунку. Ще можна виділити, що ,як і будь яка нейронна мережа, MT-LSTM потребує дуже багато вхідних даних для коректного функціонування.

Моделі з перемиканням режимів це не дуже популярний спосіб аналізу даних. Для аналізу рівня глюкози мені вдалось знайти тільки статтю Chaotic time series prediction for glucose dynamics in type 1 diabetes mellitus using regime-switching models [15] у якій досліджували різні моделі з перемиканням режимів. Ці моделі ще називають моделями з Марковськими перемиканнями [16]. Основна ідея моделей з Марковськими перемиканням в тому, що вона складається с багатьох з багатьох структур (рівнянь), які характеризують поведінку часового ряду в різних режимах. І між цими режимами переключення контролюються деякою необстежуваною змінною, яка слідує марковському процесу першого порядку. Поточне значення марковської змінної залежить лиш від минулого.

У дослідженні приймало участь 17 людей, з замірами рівня глюкози приблизно по вісім разів на день. З результатів цього дослідження у найвдалішого виду моделі вийшло дійти до точності в 93 відсотки, що є дійсно хорошим результатам. Проте передбачення обмежувались періодом в 30 хвилин

і для їх обчислення все ще потрібно відносно багато часу. Наразі ще не має публічних застосувань, які б аналізували рівень глюкози у діабетиків моделями з перемиканням режимів.

Реактивний підхід найпростіший з представлених, використовуючи його застосунки по введеному значенню одразу відображають чи здорове воно чи ні і варто було б звернутись до лікаря. Це найбільш розповсюджений вид обробки даних серед додатків для людей хворих на діабет, хоч він і використовується тільки в простих застосуваннях, а не самих використовуваних.

1.3.2 Аналіз відомих програмних продуктів

На ринку є дуже багато застосунків-довідників, які по факту є просто більш зручним замінником блокнота, не дають якогось корисного додаткового функціоналу. Більшість відомих та найбільший застосунків для контролю рівня глюкози у діабетиків аналізують рівень глюкози за допомогою нейронних мереж або реактивного підходу. Вони всі досить схожі, але кожен має якісь свої особливості, що відрізняють від конкурентів.

З таких відомих застосунків розглянемо:

- Glooko;
- Glucose Buddy;
- One Drop.

Glooko [17] – це напевно один з найбільших сервісів, який дозволяє людям хворим на діабет комплексно слідкувати за станом свого здоров’я. Він на ринку з 2011 року і як і інші SaaS (Software-as-a-Service) застосунком, яким можна користуватись як у браузері так і на мобільному додатку на платформах Android та iOs. Окрім записів рівня глюкози у додатку можна також записувати свій раціон, заняття фізичними вправами і звісно також можна записувати свої прийоми інсуліну. Додаток може показувати різні графіки та кругові діаграми для зручного відслідковування своїх значень рівня глюкози, прийомів інсуліну і тому подібне, а також надсилати повідомлення користувачеві про необхідність

прийняття інсуліну, або поміряти рівень глюкози. Хоч даний застосунок не робить передбачень, його основною перевагою перед конкурентами є сумісність з великою кількістю приладів для заміру рівня глюкози (98%), інсулінових помп (95%) та моніторів для постійного слідкування за рівнем глюкози (98%) і автоматично буде записувати значення з цих приладів до облікового запису користувача. Також даний застосунок сумісний з деякими фітнес трекерами для того, щоб автоматично підтягувати дані про фізичні вправи. Візуальний вигляд веб версії застосунку можна бачити на Рисунку 1.5.

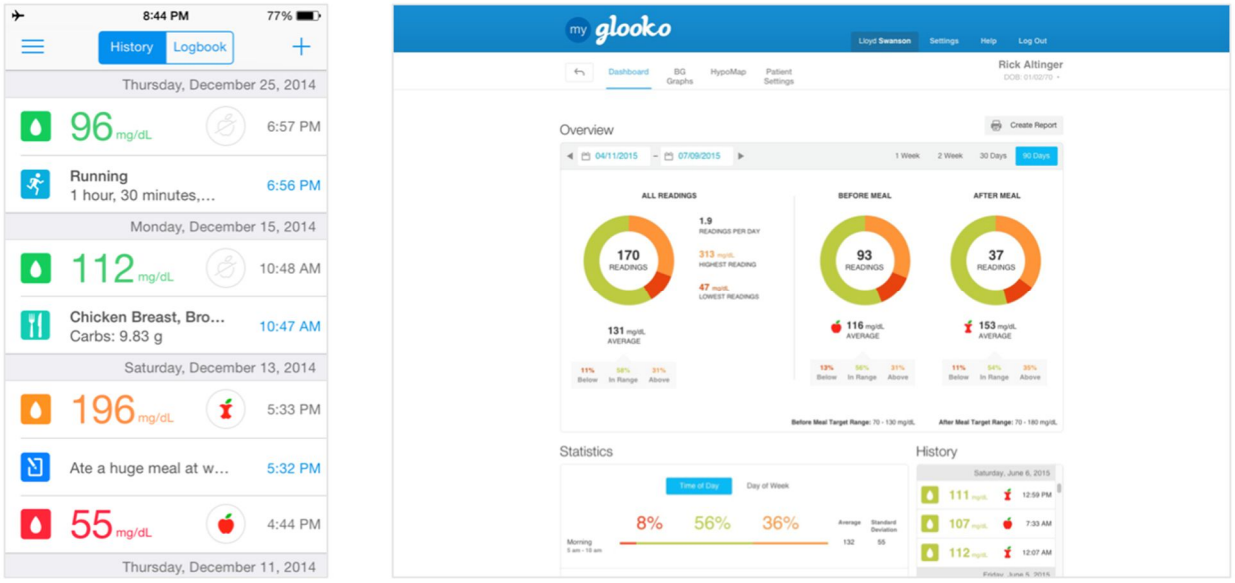


Рисунок 1.6 – Glooko

Ще один з найпопулярніших додатків для контролю своїх показників для людей хворих на діабет це Glucose Buddy [18]. Цей застосунок на ринку уже 9 років і розповсюджується також і у вигляді веб сайту і як мобільний застосунок, проте тут є явний фокус на мобільну версію. На відміну від минулого прикладу, даний застосунок не має інтегрування з будь-якими зовнішніми пристроями, але вже включає в себе підтримку деяких передбачень про можливі гіпоглікемічні та гіперглікемічні стани, хоч і не дуже точно. Звісно він також підтримує записи рівня глюкози, прийнятого інсуліну, фізичних вправ та прийомів їжі. Ще однією особливістю даного сервісу є те, що він не тільки є трекером для важливих для діабетиків даних, а ще є і соціальною мережею спеціально для людей хворих на діабет. Інтерфейс цього застосунку можна побачити на Рисунку 1.6.

Рисунок 1.7 – Glucose Buddy

Ще один цікавий сервіс це One Drop [19]. Його основна ідея в екосистемі та інтеграції з іншими своїми пристроями. Починаючи з 27 доларів на місяць людина отримує все необхідне для підтримання здоров'я: спеціальний пристрій для вимірювання рівня глюкози, який автоматично синхронізується з додатком, запаси інсуліну та інше. Цей сервіс доступний не тільки в вигляді веб версії, а і в вигляді застосунку на мобільний телефон, та навіть на годинник (Поки що підтримується тільки Apple Watch). Додаток дає можливість і власноруч записувати різноманітні дані, а також надає кожному користувачу персонального лікаря, який буде аналізувати ваші дані та давати поради про те як покращити свій стан, що їсти, які вправи і коли робити, який інсулін приймати тощо.

Рисунок 1.8 – One Drop

Як можемо побачити з цих прикладів, навіть у найпопулярніших додатках для допомоги хворим на діабет в контролі здоров'я досі не вистачає якісного передбачення гіперглікемії та гіпоглікемії. Ця функція дуже рідко зустрічається і тому розробка цього веб-застосунку є досі актуальною.

1.4 Аналіз вимог до програмного забезпечення

В системі поки що передбачається тільки одна роль: користувач. Він і буде власне використовувати наш сервіс для того, щоб записувати свої значення та стежити за своїм станом. В майбутньому можливо буде додано роль лікаря для того щоб давати більше корисної інформації та функціоналу для допомоги хворим на діабет.

Зручним і популярним способом задання вимог це діаграми використання (use-case diagrams) [20]. На рисунку 1.8 наведено діаграму використання для розроблюваної системи.

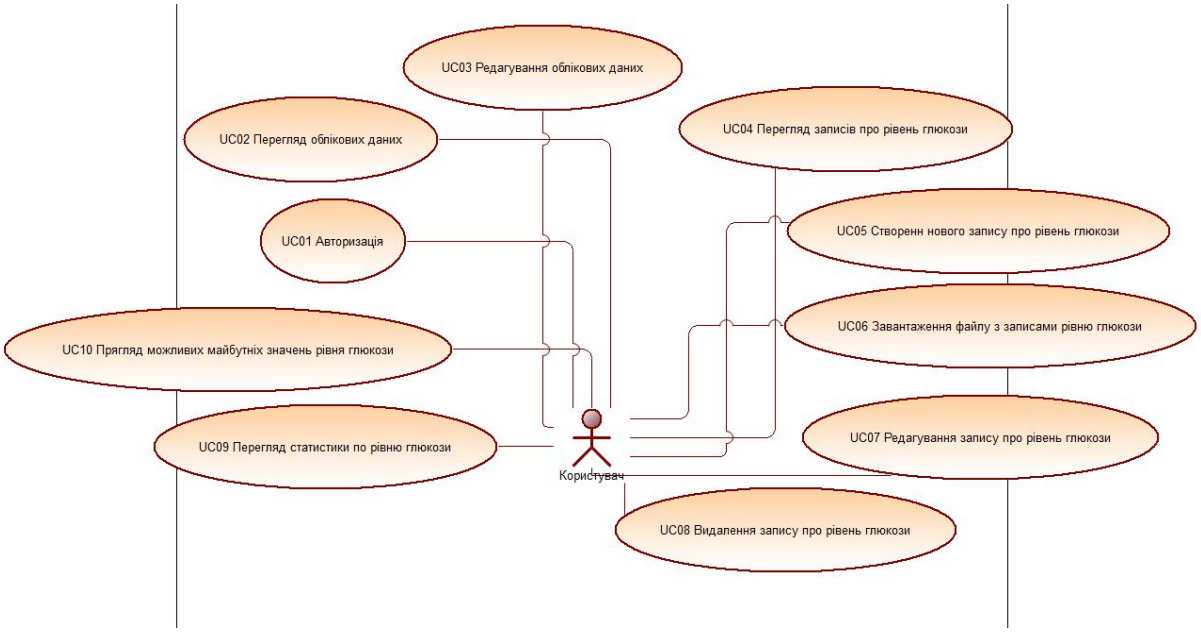


Рисунок 1.9 – Схема структура варіантів використання

Нижче наведено таблицю із ідентифікатором використання та найменуванням.

Таблиця 1.1 –Варіанти використання

Ідентифікатор	**Назва**
UC01	Авторизація
UC02	Реєстрація
UC03	Перегляд облікових даних
UC04	Редагування облікових даних
UC05	Перегляд записів про рівень глюкози
UC06	Створення нового запису про рівень глюкози
UC07	Завантаження файлу с записами рівню глюкози
UC08	Редагування запису про рівень глюкози

Продовження таблиці 1.1

UC09	Видалення запису про рівень глюкози
UC10	Перегляд статистики по рівню глюкози
UC11	Перегляд можливих майбутніх значень рівня глюкози

1.4.1 Розроблення функціональних вимог

На основі наведених вище варіантів використання було сформульовано наступні функціональні вимоги.

Таблиця 1.2 –Опис функціональних вимог

Ідентифікатор	**Опис**	**Пріоритет**
REQ01	Система має надавати можливість користувачеві реєструватись у веб-застосунку. Для цього користувач має обов’язково надати свій унікальний логін користувача, унікальну для системи адресу електронної пошти та пароль, що обмежений такими правилами: – мінімум 8 символів; – мінімум 1 літера верхнього регістру; – мінімум 1 літера нижнього регістру; – мінімум 1 цифра.	Високий
REQ02	Система має дозволяти зареєстрованому користувачу отримати авторизаційний токен, якщо він надасть наступні дані: – унікальне ім’я користувача; – пароль.	Високий

Продовження таблиці 1.2

REQ03	Система має дозволяти зареєстрованому користувачу отримати авторизаційний токен, якщо він надасть наступні дані: – унікальна електрона адреса користувача; – пароль.	Високий
REQ04	Система має надавати можливість зареєстрованому користувачу переглядати свої облікові дані, за умови, що він авторизований	Високий
REQ05	Система має надавати можливість зареєстрованому користувачу редагувати свої облікові дані, за умови, що він авторизований	Високий
REQ06	Система має надавати можливість зареєстрованому користувачу переглядати свої записи про рівень глюкози, за умови, що він авторизований	Високий
REQ07	Система має надавати можливість зареєстрованому користувачу створювати новий запис про свій рівень глюкози, за умови, що він авторизований	Високий
REQ08	Система має надавати можливість зареєстрованому користувачу завантажувати файл с записами рівню глюкози в форматі csv з полями Date (Формат MM-DD-YYYY), Time (Формат hh:mm:ss), Glucose Value (У ммоль/л) , за умови, що він авторизований	Середній

Продовження таблиці 1.2

REQ09	Система має надавати можливість зареєстрованому користувачу редагувати свої записи про рівень глюкози, за умови, що він авторизований	Високий
REQ10	Система має надавати можливість зареєстрованому користувачу видаляти свої записи про рівень глюкози, за умови, що він авторизований	Високий
REQ11	Система має надавати можливість зареєстрованому користувачу переглядати статистику по своєму рівню глюкози за день, тиждень або місяць, за умови, що він авторизований	Середній
REQ12	Система має надавати можливість зареєстрованому користувачу переглядати можливе значення глюкози через 30 хв після останнього занесеного значення, за умови, що він авторизований	Високий

Нижче наведено матрицю трасування для даного веб-застосунку.

	UC01	UC02	UC03	UC04	UC05	UC06	UC07	UC08	UC09	UC10	UC11
REQ01		■									
REQ02	■										
REQ03	■										
REQ04			■								
REQ05				■							
REQ06					■						
REQ07						■					
REQ08							■				
REQ09								■			
REQ10									■		
REQ11										■	
REQ12											■

Рисунок 1.10 – Матриця трасування

1.4.2 Розроблення нефункціональних вимог

Веб-застосування для аналізу рівня глюкози лінгвістичним методом у крові діабетиків складається із сервера та фронт-енд додатку.

Сервер надає REST API [21] для роботи з рівнем глюкози та обліковими записами користувачів. В серверну частину вбудована документація REST ендпойнтів (кінцевих точок, на адреси яких можна посилати запити та отримувати певну відповідь). Обмін даними відбувається у форматі JSON. Авторизація реалізована з використанням JSON Web Tokens [22]. Для розгортання серверної частини застосунку потрібна JVM версії 11 або вище. Усі паролі мають зберігатися у захешованому вигляді, а персональні дані користувачів чи не наявні чи зашифровані (крім адреси електронної пошти). Дані сервера мають бути ізольовані від фронт-енд частини.

Фронт-енд застосунок має працювати під браузерами Chromium версії 81.0.4044.138 або вище, Microsoft Edge версії 81.0.416.64 або вище та Safari версії 13.0 або вище.

1.4.3 Постановка комплексу завдань модулю

Основним призначенням створюваного веб-застосування є спрощення ведення записів про свій рівень глюкози людям хворим на діабет, а також надання їм можливості дізнаватись свій можливий майбутній рівень глюкози, для попередження небезпечних для життя гіпоглікемічного та гіперглікемічного станів.

Мета – застосувати набуті у процесі навчання комплексні знання різних наукових дисциплін, для побудови веб-застосування для спрощення ведення записів про рівень глюкози та надання інформації про можливий майбутній рівень глюкози згідно з наявними даними користувача.

Система має виконувати наступні задачі:

- авторизація користувача;
- реєстрація нового користувача;

– перегляд облікових даних користувача;
– редагування облікових даних користувача;
– перегляд записів про рівень глюкози поточного користувача;
– створення нового запису про рівень глюкози користувача;
– завантаження файлу з записами рівню глюкози користувача;
– редагування записів про рівень глюкози користувача;
– видалення записів про рівень глюкози користувача;
– перегляд статистики по рівню глюкози користувача;
– перегляд можливих майбутніх значень рівня глюкози користувача.

1.5 Висновки по розділу

У даному розділі були проаналізовані технологічні рішення, що використовуються для рішення поставленої задачі, та вирішують її з різною ефективністю та ресурсоємністю. Було розглянуто складові потрібні для реалізації прогнозування значень глюкози для людей хворих на діабет лінгвістичним методом. Також були описані сценарії використання, складено функціональні та нефункціональні вимоги. І найважливіше – були проаналізовані інші програмні рішення та аналоги, що зараз використовуються у даній предметній області. Проглянувши популярні рішення стало зрозуміло, що задача прогнозування у системах трекінгу для людей хворих на діабет досі не достатньо розкрита, тобто була доведена актуальність розробки даного веб-застосування.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

На рисунку 2.1 наведено BPMN діаграму [23], яка описує процес запиту можливих майбутніх значень рівня глюкози користувачів за допомогою цього веб-застосунку. Користувач робить запит використовуючи інтерфейс веб-застосунку. Застосунок реагує на цей запит, вибирає усі останні записи рівню глюкози для поточного користувача (того що надіслав запит). Отримавши з бази даних ці записи застосунок перевіряє чи відповідає частота отримання цих записів, і якщо вона не підходить ми ці дані інтерполюємо. Тепер дані, які точно відповідають потрібному об’єму, застосунок перетворює на лінгвістичну послідовність. Далі від цієї лінгвістичної послідовності береться деяка порція з кінця і порівнюється з наявними у базі лінгвістичними послідовностями. При знаходженні схожих частин, вибирається та яка зустрічається найчастіше та робиться висновок про можливе майбутнє значення рівня глюкози користувача.

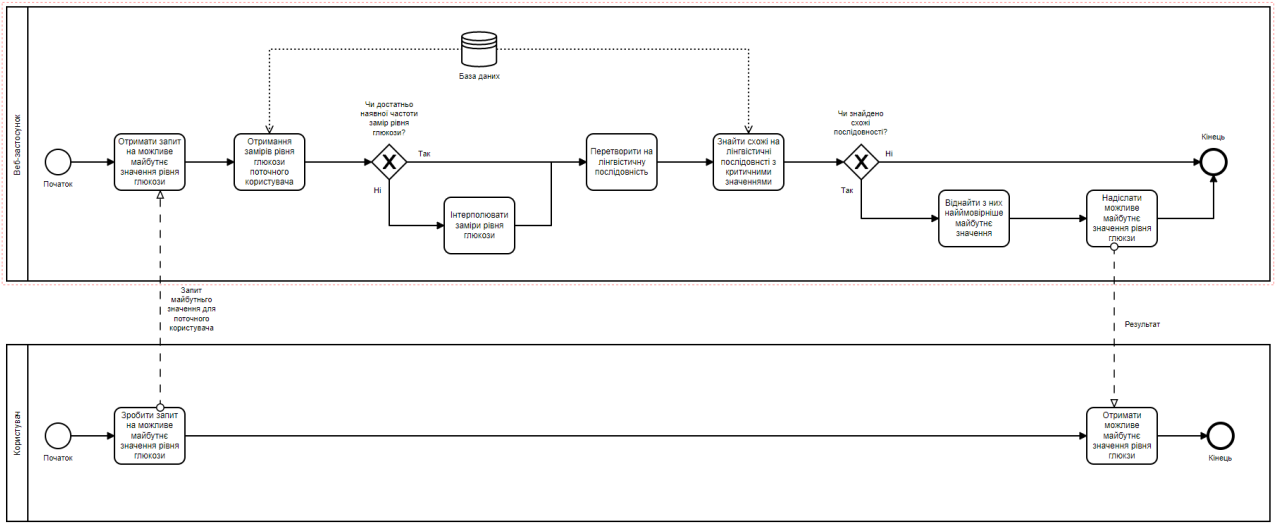


Рисунок 2.1 – Схема бізнес процесу запиту можливих майбутніх значень рівня глюкози

2.2 Архітектура програмного забезпечення

2.2.1 Опис використаних рішень

Мовою програмування для серверної частини цього веб-застосунку було обрано Java (а саме Java 13, через достатню кількість покращень та спрощень відносно найчастіше використовуваної зараз восьмої версії). У якості основної середи для розробки слугувала Intellij IDEA Ultimate Edition [24]. Так, як розроблюваний застосунок є клієнт-серверним програмним забезпеченням з точною потребою у гнучкості у процесі розробки, то для його написання було вирішено використовувати Spring Framework [25] (А точніше його версію для ще швидшої розробки Spring Boot), як найпопулярніший фреймворк загального призначення для Java.

Spring – це програмний каркас (фреймворк) з відкритим кодом, що призначений для спрощення розробки для платформи J2EE [26]. Даний фреймворк складається з великої кількості різноманітних необов’язкових модулів кожен з яких виконує якусь свою певну задачу будучи надбудовою над ядром Spring. Одними з найпопулярніших модулів є:

– Spring JDBC та Spring Data – допомагають у роботі з різноманітними базами даних, як через прямі запити до баз так і наданням можливості використання ORM (Object-Relational mapping) системи [27]. Також надають функціональність роботи з транзакціями баз даних;

– Spring MVC – даний модуль надає каркас для MVC (Model-View-Controller) структурованих застосувань та включає у себе Spring Web, що використовується для зручної роботи з HTTP запитами;

– Spring AOP – підтримка аспектоорієнтованого програмування, з використанням якого дуже зручно профілювати роботу об’єктів, або відслідковувати викликанням певних методів;

– Spring Security – надає обширну кількість функцій та анотацій для роботи з безпекою даних користувачів маючи при цьому досить прості

інтерфейси та підходи. Дуже гнучкий у конфігурації, як більшість Spring модулів.

Основною і найважливішою складовою Spring є контейнери для підтримки Inversion of Control (інверсії управління) [28]. Інверсія управління або IoC – це такий принцип побудови програми за якого її частини переважно отримують потік керування з спільної бібліотеки. Однією з найчастіше використовуваних реалізацій цього принципу є Dependency Injection [29] (Впровадження залежностей). При впровадженні залежностей об’єкти у програмі не створюють самі об’єкти від яких вони залежні, натомість ця за цю задачу в цьому випадку відповідає IoC контейнер. Саме цей підхід і використовується в Spring Framework. Саме ця можливість не тільки зробила розробку самого застосунку простішою, а і уможливила дуже легку та швидку зміну використовуваної реалізації різних сервісів, наприклад для тестування різних розподілів для побудови лінгвістичної послідовності.

Незважаючи на те, що Spring Framework не обмежує якоюсь конкретною моделлю програмування, він став надзвичайно широко поширеним в колі Java-розробників в основному як заміна і альтернатива моделі Enterprise JavaBeans [30]. Spring Framework не обмежує, а надає більше свободи Java-розробникам в проектуванні, а також він надає детально документовані і легкі у використанні засоби вирішення різноманітних проблем, що виникають при створенні додатків, як корпоративного, так і маленького масштабу.

Spring Boot – це спеціальна розробка на основі Spring Framework, яка набагато сильніше пришвидшує розробку застосувань різних розмір, суттєво зменшуючи необхідність в конфігурації. Ця надбудова над фреймворком розроблена таким чином, що додавання нових, часто використовуваних компонент програми (таких як наприклад база даних, або робота з HTTP запитами) набагато простішою та швидшою.

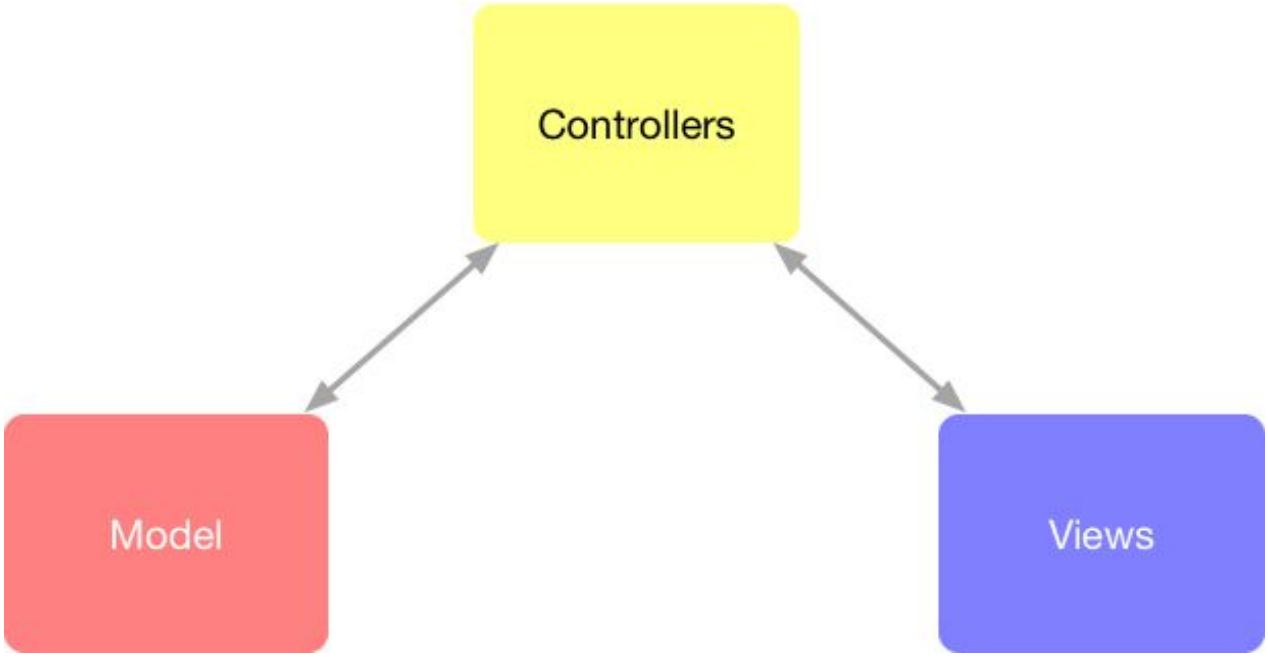


Рисунок 2.2 – Візуальне представлення використаного шаблону MVC

При побудові цього веб-застосунку, я керувався архітектурним шаблоном MVC. Model-View-Controller (Модель-вигляд-контролер) [31] – це одне з найпопулярніших архітектурних рішень, яке використовується при розробці клієнт-серверного програмного забезпечення, який поділяє систему на три взаємопов'язані частини:

– Model (Модель даних) – це шар, який буде відповідати за так звану «бізнес логіку» та роботу з сховищами даних. Тобто обробка даних, звернення до баз даних. Користувач не взаємодіє безпосередньо з цим шаром;

– View (Вигляд або ще іноді інтерфейс користувача) – це зовнішній вигляд програми, та інтерфейс за допомогою якого користувач може викликати потрібну логіку з серверної частини для досягнення своїх цілей:

– Controller (Контролер, або модуль керування) – це той шар, який сполучає модель та вигляд. Він слугує мостом, який отримує дані від користувача, передає їх до шару моделі, отримує вже оброблений результат та передає його назад для відображення користувачеві.

Таке розмежування обов'язків між шарами дозволяє правильно обмежити задачі, які будуть виконувати ці шари, зробить можливим легке розподілення

задач між різними спеціалістами, спростить розробку програмного забезпечення та зменшить можливість логічних помилок.

Для роботи цього застосунку потрібне деяке сховище даних для збереження записів рівня глюкози, даних користувача, передбачень тощо. Для цієї цілі було обрано SQL бази даних через надійність та кількість готових рішень для роботи з ними. У майбутньому для збільшення швидкодії можливо буде перейти на більш спеціалізовану NoSQL базу даних. Як СУБД було обрано PostgreSQL [32]. Це безкоштовна та потужна повноцінна реляційна база даних, що може працювати під великою кількістю операційних систем та платформ. Вона підтримує не тільки саме ядро бази даних с SQL запитами, а й мову складання виконуваного коду PL/SQL, що може бути використана наприклад для створення тригерів.

Для побудови клієнтної частини застосунку було вирішено використовувати популярний фреймворк Angular (2+) [33]. Це відкрита та безкоштовна платформа для швидкої розробки SPA на мові Typescript (також є можливою розробка на чистому JavaScript), що створений командою з компанії Google. SPA (Single-page application) або односторінковий [34] застосунок це підхід до розробки веб-застосунків при якому застосунок розміщується на одній сторінці для того, щоб надати користувачу досвід користування близький до використання настільних програм. Цей фреймворк зарекомендував себе, як хороший вибір для швидкої, гнучкої та структурованої розробки веб-застосунків різного розміру

2.2.2 Опис сховища даних

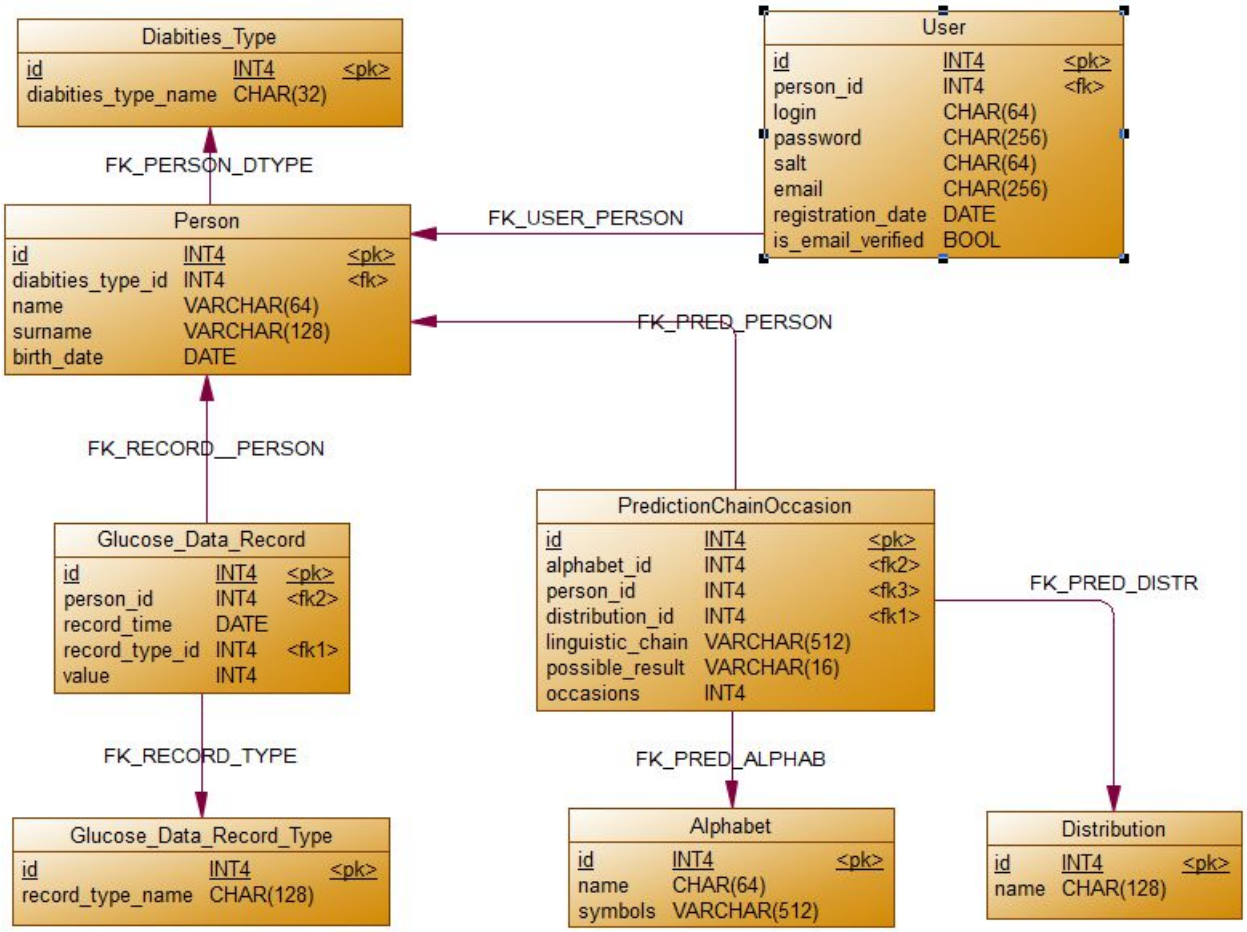


Рисунок 2.3 – Схема бази даних

Будь-який застосунок, який має зберігати дані користувачів має мати своє сховище даних. На рисунку 2.3 наведена фізична схема бази даних даного веб-застосунку. Проаналізуємо основі відношення.

User - таблиця з даними користувачів, які будуть використовуватись для її автентифікації у веб-застосунку. Хоч це відношення і відповідає за ідентифікацію користувача у сервісі, вона не є пов’язаною з іншими записами системи тож записи цієї таблиці також мають посилання на іншу сутність системи Person.

Таблиця 2.1 – Опис таблиці User

Назва поля	**Тип**	**Первинний ключ**	**Обов’язковий**	**Опис**
id	Integer	Так	Так	Ідентифікатор сутності

Продовження таблиці 2.1

person_id	Integer	Ні	Так	Ідентифікатор пов’язаної персони
login	Varchar(64)	Ні	Так	Унікальне ім’я користувача веб-застосунку
password	Varchar(256)	Ні	Так	Хеш паролю користувача
salt	Varchar(64)	Ні	Так	Сіль до пароля для SHA-256
email	Varchar(256)	Ні	Так	Адреса електронної пошти користувача
registration_date	Timestamp	Ні	Так	Дата та час реєстрації користувача
is_email_verified	Boolean	Ні	Так	Чи було підтверджено пошту користувачем?

Інше важливе для системи відношення це – Person. Саме ця сутність пов’язана з іншими в системі і відповідає за розмежування інших записів між різними людьми. Таке розділення знадобилось через можливість заносити

додаткові дані в системи не реєструючи нового користувача, та для додаткового розділення користувача, та його персональних даних.

Таблиця 2.2 – Опис таблиці Person

Назва поля	**Тип**	**Первинний ключ**	**Обов'язковий**	**Опис**
id	Integer	Так	Так	Ідентифікатор сутності
diabetes_type_id	Integer	Ні	Так	Ідентифікатор типу діабету персони
name	Varchar(64)	Ні	Ні	Ім'я персони
surname	Varchar(128)	Ні	Ні	Фамілія персони
birth_date	Timestamp	Ні	Ні	Дата народження персони

Таблиця Diabetes_Type – це відношення-словник, що визначає можливі типи діабету у людини. А саме Type 1, Type 2 та Healthy (для людей не хворих на діабет)

Таблиця 2.3 – Опис таблиці Diabetes_Type

Назва поля	**Тип**	**Первинний ключ**	**Обов'язковий**	**Опис**
id	Integer	Так	Так	Ідентифікатор сутності
diabetes_type_name	Varchar(32)	Ні	Так	Назва типу діабету

Відношення Glucose_Data_Record відповідає за збереження записів с замірами рівня глюкози кожної людини наявної у сервісі. Це потенційно таблиця з найбільшою кількістю записів та найбільшою швидкістю заповнення, по людина може вносити до 288 записів на день використовуючи пристрої частого автоматичного заміру рівня глюкози.

Таблиця 2.4 – Опис таблиці Glucose_Data_Record

Назва поля	**Тип**	**Первинний ключ**	**Обов'язковий**	**Опис**
id	Integer	Так	Так	Ідентифікатор сутності
person_id	Integer	Ні	Так	Ідентифікатор персони, якій належить запис
record_time	Timestamp	Ні	Так	Дата та час взяття заміру рівня глюкози
record_type_id	Integer	Ні	Так	Тип запису рівня глюкози
value	Integer	Ні	Так	Значення глюкози у мг/дл

Таблиця Glucose_Data_Record_Type – це відношення-словник, що відповідає за різні можливі види записів замірів рівня глюкози. В загальному там буде стояти «Автоматичний замір рівня глюкози», але можливі і варіанти такі, як «Замір рівня глюкози перед сніданком».

Таблиця 2.5 – Опис таблиці Glucose_Data_Record_Type

Назва поля	**Тип**	**Первинний ключ**	**Обов'язковий**	**Опис**
id	Integer	Так	Так	Ідентифікатор сутності
record_type_name	Varchar(128)	Ні	Так	Назва типу запису рівня глюкози

Таблиця найважливіша для аналізу рівня глюкози з метою передбачення гіпоглікемічних та гіперглікемічних станів – Prediction. Ця таблиця буде зберігати дані про знайдені лінгвістичні послідовності, що у результаті дають небезпечний для людини рівень глюкози.

Таблиця 2.6 – Опис таблиці PredictionChainOccasion

Назва поля	**Тип**	**Первинний ключ**	**Обов'язковий**	**Опис**
id	Integer	Так	Так	Ідентифікатор сутності
alphabet_id	Integer	Ні	Так	Ідентифікатор використаного у лінгвістизації алфавіту
person_id	Integer	Ні	Так	Ідентифікатор персони, у якої знайдено гіпоглікемічнмй або гіперглікемічний рівень глюкози

Продовження таблиці 2.6

distribution_id	Integer	Hi	Так	Ідентифікатор використаного у лінгвістизації розподілу
linguistic_chain	Varchar(512)	Hi	Так	Лінгвістична послідовність, що передує знайденому значенню
possible_result	Varchar(16)	Hi	Так	Літера, що відповідає знайденому значенню
occasions	Integer	Hi	Так	Кількість находжень даної комбінації символів та можливого результату для даної персони

Відношення Alphabet – це відношення-словник, що визначає можливі алфавіти, що можуть бути використані для лінгвістичного методу. Ця таблиця необхідна для швидкої та «безболісної» зміни алфавіту при потребі у майбутньому.

Таблиця 2.7 – Опис таблиці Alphabet

Назва поля	**Тип**	**Первинний ключ**	**Обов'язковий**	**Опис**

Продовження таблиці 2.7

Id	Integer	Так	Так	Ідентифікатор сутності
name	Varchar(64)	Ні	Так	Назва алфавіту
symbols	Varchar(512)	Ні	Так	Послідовність усіх символів алфавіту

Таблиця Distribution – це відношення-словник, що визначає можливі розподілення, що можуть бути використані для лінгвістичного методу. Ця таблиця, як і попередня необхідна для швидкої та «безболісної» зміни алфавіту при потребі у майбутньому.

Таблиця 2.8 – Опис таблиці Distribution

Назва поля	**Тип**	**Первинний ключ**	**Обов'язковий**	**Опис**
id	Integer	Так	Так	Ідентифікатор сутності
name	Varchar(128)	Ні	Так	Назва розподілу

2.2.3 Опис модулів коду серверної частини

Код серверної частини складається із наступних пакетів:

– com.diploma.linguistic_glucose_analyzer – базовий модуль для проекту, в нього вкладені усі інші, а також у ньому розмішений клас для розгортання сервера, та запуску Spring Boot додатку;

– com.diploma.linguistic_glucose_analyzer.constants – даний модуль містить набір сталих, що часто використовуються неодноразово по усьому проекту;

– com.diploma.linguistic_glucose_analyzer.dao – модуль, що містить у собі різноманітні інтерфейси та різні їх реалізації що доступаються до сховищ даних та маніпулюють даними, які у них наявні;

– com.diploma.linguistic_glucose_analyzer.model – модуль, що містить структури даних та сутності, що відповідають відношенням у реляційній базі даних;

– com.diploma.linguistic_glucose_analyzer.dto – модуль, який містить усі необхідні структури даних, як для «спілкування» у самому сервері, так і моделі, що надходять до сервера, та надсилаються ним же;

– com.diploma.linguistic_glucose_analyzer.service – модуль, який містить у собі все різноманітні інтерфейси та реалізації цих інтерфейсів, що відповідають за так звану бізнес-логіку додатку. Саме тут проводяться основні перетворення наявних даних;

– com.diploma.linguistic_glucose_analyzer.auth – у даному містяться усі необхідні для роботи Spring Security та у цілому реєстрації, авторизації та автентифікації класи та класи з налаштуваннями;

– com.diploma.linguistic_glucose_analyzer.service.filter – цей модуль різні фільтри та перетворювачі, що необхідні для приведення даних до потрібного вигляду;

– com.diploma.linguistic_glucose_analyzer.service.matrix – модуль, який містить у собі інтерфейси та їх реалізації для роботи з матрицею частот, для роботи з передбаченнями;

– com.diploma.linguistic_glucose_analyzer.controller – даний модуль відповідає за різноманітні реалізації контролерів та інші класи необхідні для функціонування REST API;

– com.diploma.linguistic_glucose_analyzer.utils – модуль, що містить набір допоміжних класів.

Діаграму класів веб-застосунку можна побачити у додатку Д «Графічний матеріал». Нижче, у таблиці 2.9, можна побачити опис основних класів цього програмного забезпечення.

Таблиця 2.9 – Опис основних класів та інтерфейсів

Назва під-пакету	**Назва**	**Тип**	**Опис**
–	LinguisticGlucoseAnalyzerApplication	Клас	Клас з якого стартує Spring Boot сервер з IOC контейнером та усіма залежностями
constants	LinguisticChainConstants	Інтерфейс	Інтерфейс у якому наявні усі важливі для веб-застосунку константи, що використовуються неодноразово
dao	UserDAO	Інтерфейс	Інтерфейс для об’єкту, що буде зберігати та маніпулювати користувача у сховищі даних
dao	PersonDAO	Інтерфейс	Інтерфейс для об’єкту, що буде зберігати та маніпулювати

Продовження таблиці 2.9

			даними персони у сховищі даних
dao	GlucoseDAO	Інтерфейс	Інтерфейс для об’єкту, що буде зберігати та маніпулювати даними рівню глюкози у сховищі даних
dao	GlucoseFileDAO	Інтерфейс	Інтерфейс для об’єкту, що буде обробляти файли з рівнем глюкози
dao	PredictionDAO	Інтерфейс	Інтерфейс для об’єкту, що буде зберігати та маніпулювати даними передбачень у сховищі даних
dao	AbstractGlucoseFileDAO	Абстрактний клас	Реалізація інтерфейсу GlucoseFileDAO, яка надає каркас для обробки файлів з рівнем глюкози

Продовження таблиці 2.9

dao.impl	UserDAOImpl	Клас	Базова реалізація інтерфейсу UserDAO для роботи з даними користувача
dao.impl	PersonDAOImpl	Клас	Базова реалізація інтерфейсу PersonDAO для роботи з даними персони
dao.impl	ContinuousGlucoseFileDAOImpl	Клас	Реалізація абстрактного класу AbstractGlucoseFileDAO для роботи з файлами датасету D1NAMO
dao.impl	ImMemoryGlucoseDAO	Клас	Реалізація інтерфейсу GlucoseDAO для роботи з даними рівня глюкози у пам’яті

Продовження таблиці 2.9

			програми (для тестів)
dao.impl	GlucoseDAOImpl	Клас	Базова реалізація інтерфейсу GlucoseDAO для роботи з даними рівня глюкози
dao.impl	ImMemoryPredictionDAO	Клас	Реалізація інтерфейсу PredictionDAO для роботи з даними передбачень у пам’яті програми (для тестів)
dao.impl	GlucosePredictionDAO	Клас	Базова реалізація інтерфейсу PredictionDAO для роботи з даними передбачень
model	Alphabet	Перелічуваний тип	Перелічення, яке складається з наявних алфавітів та

Продовження таблиці 2.9

			наборів символів, що їм відповідають
model	GlucoseDataCode	Перелічуваний тип	Перелічення, яке складається з можливих типів записів про заміри рівня глюкози
model	GlucoseDataRecord	Клас	Модель запису рівня глюкози, що відповідає даній у базі даних
model	Prediction	Клас	Модель передбачення, що відповідає даній у базі даних
model	User	Клас	Модель користувача, що відповідає даній у базі даних
model	Person	Клас	Модель персони, що відповідає даній у базі даних

Продовження таблиці 2.9

model	DiabetesType	Перелічуваний тип	Перелічення, яке складається з можливих значень типу діабету у персони
model	Distribution	Перелічуваний тип	Перелічення, яке складається з наявних розподілень
dto.request	LoginRequest	Клас	Клас, що відповідає JSONу, що може бути надіслано на контролер для авторизації
dto.request	SignUpRequest	Клас	Клас, що відповідає JSONу, що може бути надіслано на контролер для реєстрації
dto.response	JwtAuthentificationResponse	Клас	Клас, що відповідає JSONу, що може бути надіслано контролером для

Продовження таблиці 2.9

			авторизації користувача
dto.response	DashboardResponse	Клас	Клас, що відповідає JSONу, що може бути надіслано контролером для надання даних для побудування графіки
service	GlucoseService	Інтерфейс	Інтерфейс для об’єкту, що буде оброблювати дані записів рівня глюкози
service	LinguisticChainService	Інтерфейс	Інтерфейс для об’єкту, що буде конструювати лінгвістичні послідовності та ідентифікувати чи небезпечний рівень глюкози стоїть за певним символом
service	PredictionService	Інтерфейс	Інтерфейс для об’єкту, що буде оброблювати

Продовження таблиці 2.9

			дані передбачень
service	BadGlucoseFinderService	Інтерфейс	Інтерфейс для об’єкту, що буде знаходити записи з небезпечним для здоров’я рівнем глюкози
service	UserService	Інтерфейс	Інтерфейс для об’єкту, що буде оброблювати дані користувача
service.impl	GlucoseServiceImpl	Клас	Базова реалізація інтерфейсу GlucoseService для роботи з записами рівню глюкози
service.impl	AbstractLinguisticChainService	Абстрактний клас	Цей клас це певний каркас, що реалізує LinguisticChainService та виконує всі операції по обробці певного

Продовження таблиці 2.9

			розподілу та алфавіту
service.impl	UniformLinguisticChainService	Клас	Реалізація AbstractLinguisticChainService з рівномірним розподілом
service.impl	LogNormalLinguisticChainService	Клас	Реалізація AbstractLinguisticChainService з лог-нормальним розподілом
service.impl	PredictionServiceImpl	Клас	Базова реалізація інтерфейсу PredictionService для роботи з передбаченнями
service.impl	BadGlucoseFinderServiceImpl	Клас	Базова реалізація інтерфейсу BadGlucoseFinderService для роботи по знаходженню небезпечних рівнів глюкози

Продовження таблиці 2.9

service.impl	UserServiceImpl	Клас	Базова реалізація інтерфейсу UserService для роботи з користувачем
service.filter	RecordFilter	Інтерфейс	Інтерфейс для фільтра, що буде відсіювати непотрібні дані, або розширювати їх
service.filter	GlucoseOnlyRecordFilter	Клас	Реалізація RecordFilter, яка вибирає тільки заміри глюкози серед поданих записів
service.filter	MinMeasuresPerDayRecordFilter	Клас	Реалізація RecordFilter, яка вибирає тільки ті групи записів у яких є певна кількість замірів на день
service.filter.provider	FiltersProvider	Інтерфейс	Інтерфейс для постачальника використовуван

Продовження таблиці 2.9

			ого набору фільтрів
service.filter.provider	GlucoseFiltersProvider	Клас	Базова реалізація інтерфейсу FiltersProvider для постачання набору фільтрів
service.matrix	PredictionMatrixFactory	Інтерфейс	Інтерфейс для постачальника матриць ймовірностей отримання певного символу після деякої послідовності символів
service.matrix.model	PredictionMatrix	Клас	Модель матриці ймовірностей
service.matrix.impl	PredictionMatrixFactoryImpl	Клас	Базова реалізація PredictionMatrixFactory створення матриць ймовірностей
auth	CurrentUser	Інтерфейс	Інтерфейс анотації

Продовження таблиці 2.9

			впровадження поточного користувача системи
auth	CustomUserDetailsService	Клас	Сервіс для отримання користувача за ім’ям або ідентифікатором
auth	JwtAuthenticationEntryPoint	Клас	Клас, який є вхідною точкою для Spring Security автентифікації через JWT токени
auth	JwtAuthenticationFilter	Клас	Клас, що відповідає власно за автентифікацію з використанням Spring Security
auth	JwtTokenProvider	Клас	Клас-постачальник JWT токенів для авторизації користувача

Продовження таблиці 2.9

auth	UserPrincipal	Клас	Модель поточного користувача системи для Spring Security
controller	AuthController	Клас	Клас, що є контролером на який буде оброблювати запити, які відносяться до авторизації
controller	UserController	Клас	Клас, що є контролером на який буде оброблювати запити, які відносяться до роботи з користувачем
controller	GlucoseController	Клас	Клас, що є контролером на який буде оброблювати запити, які відносяться до роботи з

Продовження таблиці 2.9

			записами про рівень глюкози
controller	PredictionController	Клас	Клас, що є контролером на який буде оброблювати запити, які відносяться до передбачень небезпечних для людини рівнів глюкози
utils	GlucoseDataCodeUtils	Клас	Утилітарний клас, з допоміжними методами для роботи з типами записів про заміри рівня глюкози

2.3 Безпека паролю та автентифікації

При зберіганні паролю ми не можемо зберігати його у відкритому вигляді, бо тоді цю інформацію достатньо легко дістати. Щоб цього уникнути нам потрібно пропустити пароль через KDF (Key derivation function) [35], і зберігати хеш від пароля, а не сам пароль. Для цієї цілі даний веб-застосунок буде використовувати алгоритм argon2 [36]. Основними перевагами argon2 є надійність (поки що) та для її обчислення потрібно відносно багато ресурсів та обчислювальної потужності, що робить підбір паролів захешованих через argon2

нерентабельним та складним. Але недоліком argon2 є те, що який об’єм даних туди подається, стільки захешованих даних і виходить, як результат функції. Це збільшує складність обрахунків для паролів більшого розміру, але дозволяє людям з намірами зламати сервер завантажувати як пароль великі файли, тип самим ламаючи сервер.

Щоб цього уникнути ми будемо використовувати також і інший алгоритм хешування SHA-3 [37] (А саме SHA-512). Цей алгоритм і досі є надійним та його основною перевагою для нас є стала кількість символів після хешування, тобто паролі і короткі і довгі після SHA-512 будуть виглядати майже однаково. У нашому випадку ми будемо спочатку пароль пропускати через SHA-512 для того, щоб отримати хеш сталої довжини, а вже потім використаємо argon2 з цілю уникнути перебору паролів та забезпечити їх безпеку.

Наступним після реєстрації важливим процесом у веб-застосунку, безпеку якого потрібно забезпечити, є автентифікація. У результаті процесу автентифікації буде отримано токен доступу. Було прийнято рішення використовувати JWT токени, які мають вигляд JSON об’єкта та цифровому підписі. Основними складовими JWT є заголовок, сигнатура та вміст. Сигнатура обчислюється використовуючи симетричний алгоритм шифрування HS256, що використовується на послідовності з заголовку та вмісті, що поєднані розділювачем. Також, одним з важливих аргументів HS256 є секрет, який дає змогу відрізнити свою сигнатуру від чужої.

2.4 Висновки по розділу

В даному розділі було описано процес роботи запиту передбачень небезпечного для людини рівню глюкози (гіпоглікемії або гіперглікемії) з точки зору користувача та веб-застосунку.

Було описано використовувані рішення та архітектурні підходи, а також зазначено необхідність їх використання та користь від них. Також описано

використане сховище даних та показана сама фізична діаграма бази даних. Описані основні відношення та використані сутності.

Також, у даному розділі було зазначено основні модулі системи та описані класи та інтерфейси, що у них містяться з їх призначенням, що дозволить легше зрозуміти роботу цього програмного забезпечення. Знаючи цю інформацію, цей веб-застосунок можна легше покращувати та доповнювати знаючи його внутрішній устрій.

В кінці другого розділу був також розглянутий аспект безпеки даних, а саме безпека зберігання паролю та процесу автентифікації. Наведені використовувані рішення та обґрунтовано їх використання та захищеність.

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз якості ПЗ

Важливою складовою розробки та супроводу програмного забезпечення є вміння оцінювати та аналізувати його якість. Основна ідея цього аналізу це визначення відповідності певної версії програмного продукту деякому попередньо визначеному набору вимог. Ці вимоги можна розділити по відповідним характеристиками якості по ISO/IEC 25010:2011 [38] :

- функціональна придатність (Functional suitability);
- ефективність (Performance efficiency);
- сумісність (Compatibility);
- зручність (Usability);
- надійність (Reliability);
- безпека (Security);
- ремонтопридатність (Maintainability);
- переносимість (Portability).

Хоч усі ці показники є важливими, але основним показником того, що ПЗ готове до впровадження є високий процент відповідності функціональним вимогам.

3.2 Опис обсягу тестування

Метою тестування даного веб-застосування являється визначення відповідності отриманого продукту тим функціональним вимогам, що були зазначені на етапі проектування. Нижче наведено список випадків, що перевіряються:

- реєстрація нового користувача;
- реєстрація з даними уже зареєстрованого користувача;

– авторизація існуючим користувачем використовуючи унікальний логін користувача та пароль;

– авторизація існуючим користувачем використовуючи унікальну адресу електронної пошти користувача та пароль;

– авторизація з даними яких не має в системі;

– перегляд записів рівня глюкози поточного користувача;

– додавання нового запису з рівнем глюкози;

– редагування запису з рівнем глюкози;

– видалення запису з рівнем глюкози;

– перегляд можливого значення рівню глюкози через 30 хв після останнього запису рівня глюкози.

3.3 Опис контрольного прикладу

В таблицях 3.1 – 3.10 наведено інформацію, щодо проведених тестів варіантів використання.

Таблиця 3.1 – Перевірка реєстрації нового користувача

Мета тесту	Перевірка можливості реєстрації нового користувача
Відповідна URL адреса API частини	/api/auth/signup
HTTP метод	POST
Початковий стан	Користувач з даною адресою електронної пошти та логіном не наявний у системі
Вхідні дані	Обов’язкові дані користувача: унікальна адреса електронної пошти, унікальний логін та пароль, що відповідає обмеженням
Схема проведення тесту	Надати вхідні дані, та відправити запит на реєстрацію

Продовження таблиці 3.1

Очікуваний результат	Створюється новий обліковий запис користувача. Збережено до сховища даних, та заповнено введеними даними.
Стан програмного продукту після проведення випробувань	Створюється новий обліковий запис користувача. Збережено до сховища даних, та заповнено введеними даними. Надано унікальний ідентифікатор, заповнений час реєстрації та створений відповідний запис нової персони.

Таблиця 3.2 – Перевірка реєстрації з даними існуючого користувача

Мета тесту	Перевірка неможливості реєстрації нового користувача використовуючи дані існуючого користувача
Відповідна URL адреса API частини	/api/auth/signup
HTTP метод	POST
Початковий стан	Користувач з даною адресою електронної пошти та логіном наявний у системі
Вхідні дані	Обов’язкові дані користувача: унікальна адреса електронної пошти та унікальний логін наявного у системі користувача та пароль, що відповідає обмеженням
Схема проведення тесту	Надати вхідні дані, та відправити запит на реєстрацію
Очікуваний результат	Новий обліковий запис не створено, виведене повідомлення про помилку
Стан програмного продукту після проведення випробувань	Новий обліковий запис не створено, виведене повідомлення про помилку

Таблиця 3.3 – Перевірка авторизації користувача за логіном

Мета тесту	Перевірка можливості авторизації користувача за логіном
Відповідна URL адреса API частини	/api/auth/signin
HTTP метод	POST
Початковий стан	Користувач з даним логіном наявний у системі
Вхідні дані	Унікальний логін та пароль користувача
Схема проведення тесту	Надати вхідні дані, та відправити запит на авторизацію
Очікуваний результат	Користувача авторизовано
Стан програмного продукту після проведення випробувань	Користувача авторизовано, надано ідентифікаційний JWT Token

Таблиця 3.4 – Перевірка авторизації користувача за адресою електронної пошти

Мета тесту	Перевірка можливості авторизації користувача за адресою електронної пошти
Відповідна URL адреса API частини	/api/auth/signin
HTTP метод	POST
Початковий стан	Користувач з даною адресою електронної пошти наявний у системі
Вхідні дані	Унікальний логін та пароль користувача
Схема проведення тесту	Надати вхідні дані, та відправити запит на авторизацію
Очікуваний результат	Користувача авторизовано

Продовження таблиці 3.4

Стан програмного продукту після проведення випробувань	Користувача авторизовано, надано ідентифікаційний JWT Token

Таблиця 3.5 – Перевірка авторизації за даними користувача, які не наявні у системі

Мета тесту	Перевірка неможливості авторизації користувача з невірними даними
Відповідна URL адреса API частини	/api/auth/signin
HTTP метод	POST
Початковий стан	Користувач з даним логіном або адресою електронної пошти не наявний у системі
Вхідні дані	Невірні дані користувача
Схема проведення тесту	Надати вхідні дані, та відправити запит на авторизацію
Очікуваний результат	Користувача не авторизовано, виведено помилку авторизації
Стан програмного продукту після проведення випробувань	Користувача не авторизовано, виведено помилку авторизації

Таблиця 3.6 – Перевірка можливості перегляду записів рівня глюкози поточного користувача

Мета тесту	Перевірка можливості перегляду записів рівня глюкози поточного користувача
Відповідна URL адреса API частини	/api/glucose /user
HTTP метод	GET
Початковий стан	Користувач авторизований
Вхідні дані	Відсутні

Продовження таблиці 3.6

Схема проведення тесту	Відправити запит на отримання записів рівня глюкози поточного користувача
Очікуваний результат	Отримано усі записи рівня глюкози авторизованого користувача
Стан програмного продукту після проведення випробувань	Отримано усі записи рівня глюкози авторизованого користувача

Таблиця 3.7 – Перевірка можливості додавання нового запису рівня глюкози

Мета тесту	Перевірка можливості додавання нового запису рівня глюкози
Відповідна URL адреса API частини	/api/glucose
HTTP метод	POST
Початковий стан	Користувач авторизований
Вхідні дані	Тип розмірності значення рівню глюкози, значення рівню глюкози у даній розмірності, час зняття заміру
Схема проведення тесту	Надати вхідні дані та відправити запит на додання нового запису рівня глюкози
Очікуваний результат	Новий запис рівня глюкози додано
Стан програмного продукту після проведення випробувань	Новий запис рівня глюкози додано, значення рівню глюкози приведено до загального типу веб-застосунку, створено відповідний запис у сховищі даних.

Таблиця 3.8 – Перевірка можливості редагування запису рівня глюкози

Мета тесту	Перевірка можливості редагування запису рівня глюкози

Продовження таблиці 3.8

Відповідна URL адреса API частини	/api/glucose
HTTP метод	PUT
Початковий стан	Користувач авторизований, наявний хоча б один запис рівню глюкози у поточного користувача
Вхідні дані	Зміна значення рівню глюкози певного запису рівня глюкози поточного користувача
Схема проведення тесту	Надати вхідні дані та відправити запит на редагування запису рівня глюкози
Очікуваний результат	Значення рівня глюкози змінено
Стан програмного продукту після проведення випробувань	Значення рівня глюкози змінено, значення рівню глюкози приведено до загального типу веб-застосунку, змінено відповідний запис у сховищі даних.

Таблиця 3.9 – Перевірка можливості видалення запису рівня глюкози

Мета тесту	Перевірка можливості видалення запису рівня глюкози
Відповідна URL адреса API частини	/api/glucose
HTTP метод	DELETE
Початковий стан	Користувач авторизований, наявний хоча б один запис рівню глюкози у поточного користувача
Вхідні дані	Запис рівню глюкози поточного користувача, який будемо видаляти

Схема проведення тесту	Надати вхідні дані та відправити запит на видалення запису рівня глюкози
Очікуваний результат	Запис значення рівня глюкози видалено
Стан програмного продукту після проведення випробувань	Запис значення рівня глюкози видалено, , видалено відповідний запис у сховищі даних.

Таблиця 3.10 – Перевірка можливості перегляду можливого значення рівню глюкози через 30 хв після останнього запису рівня глюкози

Мета тесту	Перевірка можливості перегляду можливого значення рівню глюкози через 30 хв після останнього запису рівня глюкози
Відповідна URL адреса API частини	/api/prediction
HTTP метод	GET
Початковий стан	Користувач авторизований, наявні хоча б десять записів рівню глюкози за останній день
Вхідні дані	Відсутні
Схема проведення тесту	Відправити запит на отримання можливого значення рівню глюкози через 30 хв після останнього запису рівня глюкози
Очікуваний результат	Отриманий приблизні інтервал у якому знаходиться можливе значення рівню глюкози через 30 хв після останнього запису рівня глюкози
Стан програмного продукту після проведення випробувань	Отриманий приблизні інтервал у якому знаходиться можливе значення рівню глюкози через 30 хв після останнього.

Продовження таблиці 3.10

	запису рівня глюкози. Дані про матрицю передбачень у базі оновлено.

3.4 Висновки до розділу

В даному розділі було описано процес аналізу та тестування розроблюваного веб-застосунку. Для основних варіантів використання було наведено сценарії та результати їх тестування.

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Для розгортання серверної частини цього веб-застосунку необхідно виконати наступні дії:

– завантажити собі копію проекту за допомогою «git clone https://github.com/CatbirdBasil/linguistic_glucose_analyzer.git»;

– встановити JDK та JRE не нижче 11 версії;

– встановити Maven версії 3.6 або вище;

– встановити PostrgeSQL базу даних, або мати доступ та дані підключення до зовнішньої бази даних PostrgeSQL;

– ввести url адресу підключення до бази даних, логін та пароль адміністратора бази до файлу application.properties;

– знаходячись у корінній папці проекту виконати команду «mvn clean install»;

– виконати запуск серверного застосунку виконавши наступну команду «java -jar backend-0.01-SNAPSHOT.jar» знаходячись у папці з jar файлом застосунку.

Більшість минулих кроків можна пропустити за наявності цього jar файлу.

Для роботи з веб-сторінками застосунку необхідно мати встановленим Chromium версії 81.0.4044.138 або вище, Microsoft Edge версії 81.0.416.64 або вище чи Safari версії 13.0 або вище та зайти на адресу сайту зарезервованого тим хто хоче впровадити цей веб-застосунок.

Рекомендовано завантажувати JDK та JRE з офіційного веб-сайту компанії Oracle [39]. На рисунку 4.1 показано приклад списку доступних платформ та виконуваних файлів для інсталяції

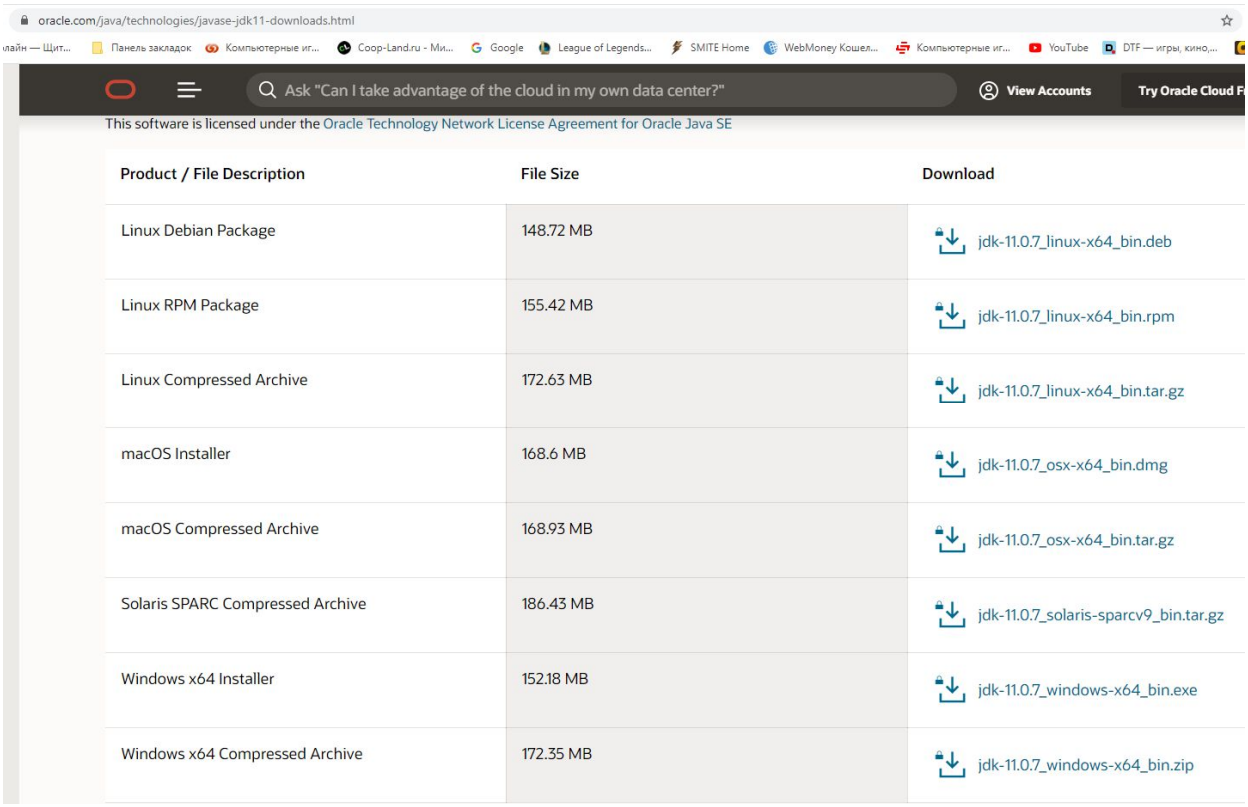


Рисунок 4.1 – Список доступних дистрибутивів JDK 11 SE

У випадку, якщо у вас вже наявна у системі інша версія Java, то потрібно, щоб вона була версії 11 вище. Інакше потрібно встановити JDK та JRE версії 11 або вище та зробити її версією за замовчуванням. Це можна зробити змінивши зміну оточення «JAVA_HOME».

За замовчуванням серверний застосунок займає порт 8080 коли піднятий. Це значення можна змінити модифікувавши змінну server.port у application.properties. Також для конфігурації у файлі application.properties доступні такі змінні:

– номер порту на якому буде розгорнуто веб-застосунок;

– секретний ключ для створення JWT Token;

– час життя одного JWT Token у мілісекундах;

– конфігурація Sl4j для основного файлу логів app.log;

– URL відключення до бази даних, логін та пароль до користувача бази даних;

```
server.port=8080

app.jwtSecret=JWTSuperSecretKey
app.jwtExpirationInMs=604800000

Logging.level.org.springframework.web=ERROR
logging.level.com.diploma.linguistic_glucose_analyzer.dao.impl.UCLGlucoseFileDAO=WARN
logging.level.com.diploma=DEBUG
logging.file.name=app.log

# Local DB settings
spring.datasource.driver-class-name=org.postgresql.Driver
spring.datasource.url=jdbc:postgresql://localhost:5432/linguistic_glucose_analyzer
spring.datasource.username=postgres
spring.datasource.password=123456
```

Рисунок 4.2 – Змінні application.properties

4.2 Робота з програмним забезпеченням

Детальну інструкцію роботи із клієнтською частиною програмного забезпечення наведено у документі «Керівництво користувача».

4.3 Супровід програмного забезпечення

Для контролю стану системи та помилок, що виникають під час її роботи можна звернутись до файлу з логами системи app.log, або при знанні внутрішньої структури системи можливо налаштувати конфігурацію Sl4j та створити свій файл, що відловлює такі логи, які потрібні під, якусь конкретну ціль

4.4 Висновки до розділу

В даному розділі було наведено інструкцію для розгортання, роботи та супроводу цього веб-застосунку.

ВИСНОВКИ

У ході виконання даної дипломної роботи було вивчено та описано предметну область застосунків для трекінгу рівня глюкози та інших даних людей хворих на діабет та підходів до передбачення рівню глюкози у крові діабетиків, проаналізовано відомі програмні продукти та показано актуальність розробки застосунку для аналізу глюкози у людей хворих на діабет з новим підходом до передбачення рівню глюкози.

Було спроектовано веб-застосунок, який складається з серверної частини та тонкого клієнта у вигляді браузерного односторінкового додатку. Серверна частина цього веб-застосунку надає API для маніпуляцій з записами рівню глюкози, такими як створення нових записів, редагування або видалення, дозволяє отримувати можливі майбутні значення рівню глюкози, а також дає можливість реєструвати та авторизувати користувачів. Для зручної взаємодії з користувачем було розроблено браузерний застосунок, який дозволяє виконувати усі ці функції зручним, не текстовим способом, а також дозволяє візуалізувати отримані дані. Так як це браузерний застосунок, то його застосування залежить не від встановленої платформи у користувача, а наявності популярного браузера.

По завершенню розробки програмного забезпечення було розроблено та виконано сценарії тестування для перевірки якісності розробленого рішенння.

Також, до цього програмного рішення було створено та надано детальну інструкцію до встановлення та розгортання, поради до супроводу, а також детальну інструкцію для користувача даного веб-застосунку.

Отже, результатом даної дипломної роботи є веб-застосунок, що надає можливості користувачам хворим на діабет у зручній формі слідкувати за своїм рівнем глюкози, бачити динаміку змін у цьому рівні, а також отримувати можливі майбутні значення рівню глюкози, що потенційно може врятувати людину від небезпечних для здоров’я гіпоглікемічних та гіперглікемічних станів.

ПЕРЕЛІК ПОСИЛАНЬ

1) Олищук Андрей Владимирович Разработка Web-приложений на PHP 5. Профессиональная работа. — М.: «Вильямс», 2006. — С. 352. — ISBN 5-8459-0944-9.

2) The open D1NAMO dataset: A multi-modal dataset for research on non-invasive type 1 diabetes management [Електронний ресурс] / [F. Dubosson, J. Ranvier, S. Bromuri та ін.] // Informatics in Medicine Unlocked. – 2018. – Режим доступу до ресурсу: https://doi.org/10.1016/j.imu.2018.09.003.

3) Kahn M. UCI Diabetes Data Set [Електронний ресурс] / Michael Kahn. – 1998. – Режим доступу до ресурсу: https://archive.ics.uci.edu/ml/datasets/diabetes.

4) Вержбицкий В. М.. Основы численных методов: учебник для вузов. — М. : Высшая школа, 2002. — 840 с. — ISBN 5-06-004020-8.

5) Волков Е. А. Глава 1. Приближение функций многочленами. § 11. Сплайны // Численные методы. — Учеб. пособие для вузов. — 2-е изд., испр.. — М.: Наука, 1987. — С. 63-68. — 248 с.

6) Fu K. S., Sequential Methods in Pattern Recognition and Machine Learning, Academic Press, 1968.

7) Баклан І. В. Лінгвістичне моделювання: основи, методи, деякі прикладні аспекти / І. В. Баклан // Систем. технології. — 2011. — № 3. — С. 10-19.

8) Coded Character Sets, History and Development — 1. — Addison-Wesley Publishing Company, Inc., 1980. — P. 6, 66, 211, 215, 217, 220, 223, 228, 236—238, 243—245, 247—253, 423, 425—428, 435—439. — ISBN 978-0-201-14460-4.

9) Ширяев, А. Н. Вероятность. — М.: Наука, 1980. — С. 45, 166.

10) Young G. A. Essentials of Statistical Inference / G. A. Young, R. L. Smith. – Cambridge: Cambridge University Press, 2005. – (Cambridge Series in Statistical and Probabilistic Mathematics; 16).

11) Patel J. Handbook of the Normal Distribution, Second Edition / J. Patel, C. Read., 1996. – (Statistics: A Series of Textbooks and Monographs; 150).

12) Log Normal Distribution [Електронний ресурс] // MathWorld – Режим доступу до ресурсу: https://mathworld.wolfram.com/LogNormalDistribution.html.

13) Predicting Blood Glucose Dynamics with Multi-time-series Deep Learning [Електронний ресурс] // Proceedings of SenSys' 17. – 2017. – Режим доступу до ресурсу: https://www.researchgate.net/publication/322548037_Predicting_Blood_Glucose_Dynamics_with_Multi-time-series_Deep_Learning.

14) A Bi-directional Multiple Timescales LSTM Model for Grounding of Actions and Verbs [Електронний ресурс] / A.Antunes, A. Laflaquiere, T. Ogata, A. Cangelosi // IEEE. – 2019. – Режим доступу до ресурсу: https://doi.org/10.1109/IROS40897.2019.8967799.

15) Frandes M. Chaotic time series prediction for glucose dynamics in type 1 diabetes mellitus using regime-switching models [Електронний ресурс] / M. Frandes, B. Timar, R. Timar // Sci Rep. – 2017. – Режим доступу до ресурсу: https://doi.org/10.1038/s41598-017-06478-4.

16) Асимптотика нормованого керування з марковськими перемиканнями / А. В. Нікітін, У. Т. Хімка // Український математичний журнал. - 2016. - Т. 68, № 8. - С. 1092-1101. - Режим доступу: http://nbuv.gov.ua/UJRN/UMJ_2016_68_8_10

17) Glooko [Електронний ресурс] – Режим доступу до ресурсу: https://www-int.glooko.com/.

18) Glucose Buddy [Електронний ресурс] – Режим доступу до ресурсу: https://www.glucosebuddy.com/.

19) One Drop [Електронний ресурс] – Режим доступу до ресурсу: https://onedrop.today/.

20) Rumbaugh J. The unified modeling language reference manual / J. Rumbaugh, I. Jacobson, G. Booch. – Reading, Mass: Addison-Wesley, 1999. – (1).

21) 21. Richardson L. RESTful Web APIs / L. Richardson, M. Amundsen, S. Ruby., 2013. – (O'Reilly). – (1).

22) JSON Web Tokens [Електронний ресурс] – Режим доступу до ресурсу: https://jwt.io/.

23) Business Process Model and Notation (BPMN) [Електронний ресурс]. – 1997. – Режим доступу до ресурсу: http://www.bpmn.org/.

24) Intellij IDEA Ultimate Edition [Електронний ресурс] – Режим доступу до ресурсу:
https://www.jetbrains.com/ru-ru/idea/.

25) Pro Spring 5: An In-Depth Guide to the Spring Framework and Its Tools / I.Cosmina, R. Harrop, C. Schaefer, C. Ho., 2019. – 1120 с.

26) Differences between Java EE and Java SE [Електронний ресурс] – Режим доступу до ресурсу: https://docs.oracle.com/javaee/6/firstcup/doc/gkhoy.html.

27) Ноубл, Дж., Андерсон, Т., Брэйтуэйт, Г., Казарио, М., Третола, Р. Flex 4. Рецепты программирования. — БХВ-Петербург, 2011. — С. 548. — 720 с.

28) Fowler M. InversionOfControl [Електронний ресурс] / Martin Fowler – Режим доступу до ресурсу: https://martinfowler.com/bliki/InversionOfControl.html.

29) Fowler M. Inversion of Control Containers and the Dependency Injection pattern [Електронний ресурс] / Martin Fowler – Режим доступу до ресурсу: https://www.martinfowler.com/articles/injection.html.

30) Enterprise JavaBeans Technology [Електронний ресурс] – Режим доступу до ресурсу: https://www.oracle.com/technetwork/java/index-jsp-140203.html.

31) Рогачев С. Обобщенный Model-View-Controller [Електронний ресурс] / Сергей Рогачев // Филиал ООО "Лукойл-Информ". – 2007. – Режим доступу до ресурсу: http://rsdn.org/article/patterns/generic-mvc.xml.

32) PostgreSQL [Електронний ресурс] – Режим доступу до ресурсу: https://www.postgresql.org/docs/current/history.html.

33) Angular [Електронний ресурс] – Режим доступу до ресурсу: https://angular.io/.

34) Single-page applications vs. multiple-page applications: pros, cons, pitfalls - BLAKIT - IT Solutions. // BLAKIT - IT Solutions. – 2017.

35) Bezzi M. Data privacy / Michele Bezzi., 2011. – (Privacy and Identity Management for Life).

36) Christian Forler, Eik List, Stefan Lucks, Jakob Wenzel. Overview of the Candidates for the Password Hashing Competition and their resistance against Garbage-Collector Attacks. — Technology and Practice of Passwords, 2015. — С. 3—18.

37) Shanghai Cooperation Organisation To Introduce ‘Mutual Settlement In National Currencies’ And Ditch US Dollar [Електронний ресурс] // Silk Road Briefing. – 2020. – Режим доступу до ресурсу: https://www.silkroadbriefing.com/news/2020/03/18/shanghai-cooperation-organisation-introduce-mutual-settlement-national-currencies-ditch-us-dollar/.

38) ISO/IEC 25010:2011 Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models [Електронний ресурс]. – 2011. – Режим доступу до ресурсу: https://www.iso.org/standard/35733.html.

39) Oracle [Електронний ресурс] – Режим доступу до ресурсу: https://www.oracle.com/ru/index.html.

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

“___” _____ 2020 р.

Веб-застосування для аналізу рівня глюкози у крові діабетиків лінгвістичним методом

Технічне завдання

КПІ.ІП-6126.045440.02.91

“ПОГОДЖЕНО”

Керівник проєкту:

_____ О.К. Очеретяний

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____І.О. Шаверський

Київ – 2020 року

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Веб-застосування для аналізу рівня глюкози у крові діабетиків лінгвістичним методом

Галузь застосування: Мережа Інтернет та галузь охорони здоров’я

Наведене технічне завдання поширюється на розробку Веб-застосування для аналізу рівня глюкози лінгвістичним методом у крові діабетиків [045440], котре використовується для аналізу загальнодоступних медичних даних і на його основі проведення передбачення для кінцевих користувачів і попередженню їх про можливі ризики значного збільшення чи зменшення глюкози (гіперглікемія та гіпоглікемія).

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки WEB-застосунку для прогнозування рівня глюкози є завдання на дипломне проектування, затверджене кафедрою автоматизованих систем обробки інформації та управління Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» («КПІ ім. Ігоря Сікорського»).

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Призначенням розробки даного програмного продукту є аналіз показників цукру в крові людей хворих на діабет.

Аналіз в майбутньому надасть можливості для предиктивного визначення граничних станів (гіпоглікемії та гіперглікемії) для їх попередження. Тобто попередження людей про можливі ризики для їх здоров’я спонукання їх до контролю і введення даних про їх поточний рівень глюкози.

В майбутньому дану базу даних при згоді користувача можна буде використовувати в медичних дослідженнях діабету.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

4.1.1 Програмне забезпечення повинно забезпечувати виконання наступних основних функції:

4.1.1.1 Для користувача:

- авторизація;
- реєстрація;
- перегляд облікових даних;
- редагування облікових даних;
- перегляд записів про рівень глюкози;
- створення нового запису про рівень глюкози;
- завантаження файлу з записами рівню глюкози;
- редагування записів про рівень глюкози;
- видалення записів про рівень глюкози;
- перегляд статистики по рівню глюкози;
- перегляд можливих майбутніх значень рівня глюкози.

4.1.2 Розробка проводилась на платформі Windows, але серверну частину веб застосунку можна розгортати на платформі Linux, за умови що на ній будуть встановлені усі передбачені програми та залежності

4.1.3 Додаткові вимоги.

Не передбачені.

4.2 Вимоги до надійності

4.2.1 Передбачити контроль введення інформації

4.2.2 Передбачити захист від некоректних дій користувача

4.2.3 Передбачити уникнення втрат персональних даних користувачів

4.2.4 Забезпечити цілісність інформації в базі даних

4.3 Умови експлуатації

4.3.1 Умови експлуатації згідно СанПін 2.2.2.542 – 96

Не висуваються

4.3.2 Обслуговування

Окрім встановлення та запуску серверної частини додатку обслуговування не передбачається.

4.4 Вимоги до складу і параметрів технічних засобів

4.4.1 Програмне забезпечення повинно функціонувати на IBM-сумісних персональних комп‘ютерах

4.4.2 Мінімальна конфігурація технічних засобів

4.4.2.1 Тип процесор

64-розрядний процесор з тактовою частотою не нижче 1.4 ГГц

4.4.2.2 Об‘єм ОЗП

Не менше 2 ГБ

4.4.2.3 Об’єм жорсткого диску

Не менше 20 ГБ.

4.5 Вимоги до інформаційної та програмної сумісності

а) Програмне забезпечення повинно працювати під управлінням операційних систем сімейства WIN64 (Windows 7, Windows 8 та Windows 10) або Unix.

б) Вхідні дані повинні бути представлені в наступному форматі: Вхідні дані, які будуть приходити на сервер з клієнтської частини нашого WEB-додатку мають мати формат JSON (JavaScript Object Notation).

в) Вихідні дані повинні бути представлені в наступному форматі: Вихідні дані, які будуть приходити на клієнтську частини нашого WEB-додатку з сервера мають мати формат JSON (JavaScript Object Notation).

г) Програмне забезпечення повинно працювати під управлінням операційних систем Windows XP з пакетом оновлення 2 +, Windows Vista, Windows 7, Windows 8, Windows 10 або новіших, Mac OS X 10 або новіша, Ubuntu 10.04 +, Fedora Linux 14, Debian 6 +, OpenSuSE 11.3 +.

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не висуваються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

4.8 Спеціальні вимоги

Має бути створена установча версію програмного забезпечення.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Попередній склад програмної документації

а) Супроводжувальна документація

1) Пояснювальна записка.

2) Технічне завдання.

3) Керівництво користувача.

4) Програма та методика тестування.

б) Довідникова документація

1) Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

2) Програмне забезпечення повинно мати довідникову систем.

в) Графічна документація

1) Схема структурна бази даних.

2) Креслення екранних форм.

3) Схема структурна варіантів використання.

4) Схема структурна класів програмного забезпечення.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

№	Назва етапу	Строк,	Звітність
1.	Вивчення літератури за тематикою проекту	28.02.2020	
2.	Розробка технічного завдання	15.03.2020	Технічне завдання
3.	Аналіз вимог та уточнення специфікацій	01.04.2020	Специфікації програмного забезпечення
4.	Проектування структури програмного забезпечення, проектування компонентів	01.05.2020	Схема структурна програмного забезпечення та специфікація компонентів
5.	Програмна реалізація програмного забезпечення	02.05.2020	Тексти програмного забезпечення
6.	Тестування програмного забезпечення	13.05.2020	Тести, результати тестування
7.	Розробка матеріалів текстової частини проекту	16.05.2020	Пояснювальна записка.
8.	Розробка матеріалів графічної частини проекту	18.05.2020	Графічний матеріал проекту
9.	Оформлення технічної документації проекту	20.05.2020	Технічна документація

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

7.1 Види випробувань

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

“___” _____ 2020 р.

Веб-застосування для аналізу рівня глюкози у крові діабетиків лінгвістичним методом

Опис програми

КПІ.ІП-6126.045440.03.13

“ПОГОДЖЕНО”

Керівник проєкту:

_____ О.К. Очеретяний

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____І.О. Шаверський

Київ – 2020 року

Тексти програмного коду

Веб-застосування для аналізу рівня глюкози лінгвістичним методом у крові діабетиків

(Найменування програми (документа))

DVD-R

(Вид носія даних)

32 арк, 13 Мб

(Обсяг програми (документа) , арк.,) Кб)

Київ – 2020

```java
package com.diploma.linguistic_glucose_analyzer;

import com.diploma.linguistic_glucose_analyzer.constants.LinguisticChainConstants;
import org.hibernate.SessionFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.EnableAutoConfiguration;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConfiguration;
import org.springframework.boot.autoconfigure.jdbc.DataSourceTransactionManagerAutoConfiguration;
import org.springframework.boot.autoconfigure.orm.jpa.HibernateJpaAutoConfiguration;
import org.springframework.context.annotation.Bean;
import org.springframework.core.env.Environment;
import org.springframework.jdbc.datasource.DriverManagerDataSource;
import org.springframework.orm.hibernate5.HibernateTransactionManager;
import org.springframework.orm.hibernate5.LocalSessionFactoryBean;

import javax.annotation.PostConstruct;
import javax.sql.DataSource;
import java.util.Properties;
import java.util.TimeZone;

@SpringBootApplication(exclude = { //
    DataSourceAutoConfiguration.class, //
    DataSourceTransactionManagerAutoConfiguration.class, //
    HibernateJpaAutoConfiguration.class })
public class LinguisticGlucoseAnalyzerApplication {

  @PostConstruct
  void init() {
    TimeZone.setDefault(TimeZone.getTimeZone(LinguisticChainConstants.DEFAULT_ZONE));
  }

  @Autowired
  private Environment env;

  public static void main(String[] args) {
    SpringApplication.run(LinguisticGlucoseAnalyzerApplication.class, args);
  }

  @Bean(name = "dataSource")
  public DataSource getDataSource() {
    DriverManagerDataSource dataSource = new DriverManagerDataSource();

    // See: application.properties
    dataSource.setDriverClassName(env.getProperty("spring.datasource.driver-class-name"));
    dataSource.setUrl(env.getProperty("spring.datasource.url"));
    dataSource.setUsername(env.getProperty("spring.datasource.username"));
    dataSource.setPassword(env.getProperty("spring.datasource.password"));

    System.out.println("## getDataSource: " + dataSource);

    return dataSource;
  }

  @Autowired
  @Bean(name="entityManagerFactory")
  public SessionFactory getSessionFactory(DataSource dataSource) throws Exception {
    Properties properties = new Properties();

    // See: application.properties
    properties.put("hibernate.dialect", env.getProperty("spring.jpa.properties.hibernate.dialect"));
    properties.put("hibernate.show_sql", env.getProperty("spring.jpa.show-sql"));
    properties.put("current_session_context_class", //
        env.getProperty("spring.jpa.properties.hibernate.current_session_context_class"));
```

```
        // Fix Postgres JPA Error:
        // Method org.postgresql.jdbc.PgConnection.createClob() is not yet implemented.
        // properties.put("hibernate.temp.use_jdbc_metadata_defaults",false);

        LocalSessionFactoryBean factoryBean = new LocalSessionFactoryBean();

        // Package contain entity classes
        factoryBean.setPackagesToScan(new String[] { "com.diploma.linguistic_glucose_analyzer.model" });
        factoryBean.setDataSource(dataSource);
        factoryBean.setHibernateProperties(properties);
        factoryBean.afterPropertiesSet();
        //
        SessionFactory sf = factoryBean.getObject();
        System.out.println("## getSessionFactory: " + sf);
        return sf;
    }

    @Autowired
    @Bean(name = "transactionManager")
    public HibernateTransactionManager getTransactionManager(SessionFactory sessionFactory) {
        HibernateTransactionManager transactionManager = new HibernateTransactionManager(sessionFactory);

        return transactionManager;
    }

}
```

```
package com.diploma.linguistic_glucose_analyzer.model;

import lombok.Getter;

@Getter
public enum Alphabet {
    ENGLISH(1L, "ABCDEFGHIJKLMNOPQRSTUVWXYZ".toCharArray()),
    ASCII100(2L, Alphabet.generateCharArray(26, 126));

    private final long id;
    private final char[] symbols;

    Alphabet(long id, char[] symbols) {
        this.id = id;
        this.symbols = symbols;
    }

    public static char[] generateCharArray(int start, int end) {
        int size = end - start;

        char[] result = new char[size];

        for (int i = 0; i < size; i++) {
            result[i] = (char) (i + start);
        }

        return result;
    }

    public static Alphabet valueOf(long id) {
        for (Alphabet alphabet : values()) {
            if (alphabet.id == id) {
                return alphabet;
            }
        }

        return null;
    }
}
```

```
package com.diploma.linguistic_glucose_analyzer.model;

import lombok.Getter;

@Getter
public enum DiabetesType {
   TYPE_ONE(1L),
   TYPE_TWO(2L),
   NO_DIABETES(3L);

   private final long id;

   DiabetesType(long id) {
      this.id = id;
   }

   public static DiabetesType valueOf(long id) {
      for (DiabetesType type : values()) {
         if (type.id == id) {
            return type;
         }
      }

      return null;
   }
}
```

```
package com.diploma.linguistic_glucose_analyzer.model;

import lombok.Getter;

@Getter
public enum Distribution {
   UNIFORM(1L),
   LOG_NORMAL(2L),
   PARABOLIC(3L);

   private final long id;

   Distribution(long id) {
      this.id = id;
   }

   public static Distribution valueOf(long id) {
      for (Distribution distr : values()) {
         if (distr.id == id) {
            return distr;
         }
      }

      return null;
   }
}
```

```
package com.diploma.linguistic_glucose_analyzer.model;

import lombok.Getter;

@Getter
public enum GlucoseDataCode {
   REGULAR_INSULIN_DOSE(33L),
   NPH_INSULIN_DOSE(34L),
   ULTRALENTE_INSULIN_DOSE(35L),
```

```
    UNSPECIFIED_BLOOD_GLUCOSE_MEASUREMENT(48L),
    UNSPECIFIED_BLOOD_GLUCOSE_MEASUREMENT_2(57L),
    PRE_BREAKFAST_BLOOD_GLUCOSE_MEASUREMENT(58L),
    POST_BREAKFAST_BLOOD_GLUCOSE_MEASUREMENT(59L),
    PRE_LAUNCH_BLOOD_GLUCOSE_MEASUREMENT(60L),
    POST_LAUNCH_BLOOD_GLUCOSE_MEASUREMENT(61L),
    PRE_SUPPER_BLOOD_GLUCOSE_MEASUREMENT(62L),
    POST_SUPPER_BLOOD_GLUCOSE_MEASUREMENT(63L),
    PRE_SNACK_BLOOD_GLUCOSE_MEASUREMENT(64L),
    HYPOGLYCEMIC_SYMPTOMS(65L),
    TYPICAL_MEAL_INGESTION(66L),
    MORE_THAN_USUAL_MEAL_INGESTION(67L),
    LESS_THAN_USUAL_MEAL_INGESTION(68L),
    TYPICAL_EXERCISE_ACTIVITY(69L),
    MORE_THAN_USUAL_EXERCISE_ACTIVITY(70L),
    LESS_THAN_USUAL_EXERCISE_ACTIVITY(71L),
    UNSPECIFIED_SPECIAL_EVENT(72L),
    UNKNOWN_CODE(-1L);

    private final long code;

    GlucoseDataCode(long code) {
        this.code = code;
    }

    public static GlucoseDataCode valueOf(long code) {
        for (GlucoseDataCode codeValue: values()) {
            if (codeValue.code == code) {
                return codeValue;
            }
        }

        return UNKNOWN_CODE;
    }
}


package com.diploma.linguistic_glucose_analyzer.model;

import com.fasterxml.jackson.annotation.JsonIgnore;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.*;
import java.time.Instant;

@Data
@NoArgsConstructor
@Entity
@Table(name = "Glucose_Data_Record")
public class GlucoseDataRecord {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id", nullable = false)
    private long id;
//
//    @Column(name = "person_id", nullable = false)
//    private long personId;

    @JsonIgnore
    @OneToOne(fetch = FetchType.EAGER)
    @JoinColumn(name = "person_id")
    private Person person;
```

```
    @Column(name = "record_time", nullable = false)
    private Instant eventTime; //TODO Possible problems with date saving to db

    @Column(name = "record_type_id", nullable = false)
    private long codeId;

    @Column(name = "value", nullable = false)
    private int value;

    public GlucoseDataRecord(Instant eventTime, GlucoseDataCode code, int value) {
        this.eventTime = eventTime;
        setCode(code);
        this.value = value;
    }

    public GlucoseDataCode getCode() {
        return GlucoseDataCode.valueOf(codeId);
    }

    public void setCode(GlucoseDataCode code) {
        codeId = code.getCode();
    }
}


package com.diploma.linguistic_glucose_analyzer.model;

import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.*;
import java.sql.Timestamp;
import java.time.Instant;

@Data
@Entity
@NoArgsConstructor
@Table(name = "Person")
public class Person {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id", nullable = false)
    private long id;

    @Column(name = "diabetes_type_id", nullable = false)
    private long diabetesTypeId;

    @Column(name = "name", length = 64)
    private String name;

    @Column(name = "surname", length = 128)
    private String surname;

    @Column(name = "birth_date")
    private Instant birthDate;

    public Person(long diabetesTypeId) {
        this.diabetesTypeId = diabetesTypeId;
    }

    public Person(long diabetesTypeId, String name, String surname, Instant birthDate) {
        this.diabetesTypeId = diabetesTypeId;
        this.name = name;
        this.surname = surname;
        this.birthDate = birthDate;
```

```
    }

    public Person(DiabetesType diabetesType) {
        this(diabetesType.getId());
    }

    public Person(DiabetesType diabetesType, String name, String surname, Instant birthDate) {
        this(diabetesType.getId(), name, surname, birthDate);
    }

    public DiabetesType getDiabetesType() {
        return DiabetesType.valueOf(diabetesTypeId);
    }

    public void setDiabetesType(DiabetesType type) {
        diabetesTypeId = type.getId();
    }
}
```

```
package com.diploma.linguistic_glucose_analyzer.model;

import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.*;

@Data
@NoArgsConstructor
@Entity
@Table(name = "Prediction_Chain_Occasion")
public class PredictionChainOccasion {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id", nullable = false)
    private long id;

    @Column(name = "alphabet_id", nullable = false)
    private long alphabetId;

    @Column(name = "distribution_id", nullable = false)
    private long distributionId;
//
//    @Column(name = "person_id", nullable = false)
//    private long personId;

    @OneToOne(fetch = FetchType.EAGER)
    @JoinColumn(name = "person_id")
    private Person person;

    @Column(name = "linguistic_chain", nullable = false)
    private String linguisticChain;

    @Column(name = "possible_result", nullable = false)
    private String possibleResult;

    @Column(name = "occasions", nullable = false)
    private long occasions;

    public Alphabet getAlphabet() {
        return Alphabet.valueOf(alphabetId);
    }
```

```
    public void setAlphabet(Alphabet alphabet) {
        alphabetId = alphabet.getId();
    }

    public Distribution getDistribution() {
        return Distribution.valueOf(distributionId);
    }

    public void setDistribution(Distribution distribution) {
        distributionId = distribution.getId();
    }
}
```

```
package com.diploma.linguistic_glucose_analyzer.model;

public enum ProblemType {
    HYPOGLYCEMIA,
    HYPERGLYCEMIA
}
```

```
package com.diploma.linguistic_glucose_analyzer.model;

import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.*;
import java.sql.Timestamp;

@Data
@Entity
@NoArgsConstructor
@Table(name = "`User`")
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id", nullable = false)
    private long id;

    @OneToOne(fetch = FetchType.EAGER)
    @JoinColumn(name = "person_id")
    private Person person;

    @Column(name = "login", nullable = false)
    private String login;

    @Column(name = "password", nullable = false)
    private String password;

    @Column(name = "salt", nullable = false)
    private String salt;

    @Column(name = "email", nullable = false)
    private String email;

    @Column(name = "registration_date", nullable = false)
    private Timestamp registrationDate;

    @Column(name = "is_email_verified", nullable = false)
    private Boolean isEmailVerified;
```

```
    public User(Person person, String login, String password, String email, Timestamp registrationDate) {
        this.person = person;
        this.login = login;
        this.password = password;
        this.email = email;
        this.registrationDate = registrationDate;
        this.isEmailVerified = false;
        this.salt = "";
    }
}
```

```
package com.diploma.linguistic_glucose_analyzer.config;

import com.diploma.linguistic_glucose_analyzer.auth.CustomUserDetailsService;
import com.diploma.linguistic_glucose_analyzer.auth.JwtAuthenticationEntryPoint;
import com.diploma.linguistic_glucose_analyzer.auth.JwtAuthenticationFilter;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.http.HttpMethod;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.config.BeanIds;
import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;
import org.springframework.security.config.annotation.method.configuration.EnableGlobalMethodSecurity;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;
import org.springframework.security.config.http.SessionCreationPolicy;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;

@Configuration
@EnableWebSecurity
@EnableGlobalMethodSecurity(
    securedEnabled = true,
    jsr250Enabled = true,
    prePostEnabled = true
)
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    @Autowired
    private CustomUserDetailsService customUserDetailsService;

    @Autowired
    private JwtAuthenticationEntryPoint unauthorizedHandler;

    @Bean
    public JwtAuthenticationFilter jwtAuthenticationFilter() {
        return new JwtAuthenticationFilter();
    }

    @Bean(BeanIds.AUTHENTICATION_MANAGER)
    @Override
    public AuthenticationManager authenticationManagerBean() throws Exception {
        return super.authenticationManagerBean();
    }

    @Bean
    public PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }

    @Override
    public void configure(AuthenticationManagerBuilder authenticationManagerBuilder) throws Exception {
        authenticationManagerBuilder
```

```
            .userDetailsService(customUserDetailsService)
            .passwordEncoder(passwordEncoder());
    }

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http
            .cors()
            .and()
            .csrf()
            .disable()
            .exceptionHandling()
            .authenticationEntryPoint(unauthorizedHandler)
            .and()
            .sessionManagement()
            .sessionCreationPolicy(SessionCreationPolicy.STATELESS)
            .and()
            .authorizeRequests()
            .antMatchers("/",
                "/favicon.ico",
                "/**/*.png",
                "/**/*.gif",
                "/**/*.svg",
                "/**/*.jpg",
                "/**/*.html",
                "/**/*.woff2",
                "/**/*.css",
                "/**/*.js")
            .permitAll()
            .antMatchers("/api/auth/**", "/h2/**")
            .permitAll()
            .antMatchers(HttpMethod.GET, "/api/auth/**", "/api/users/**")
            .permitAll()
            .anyRequest()
            .authenticated();

        http.headers().frameOptions().disable();

        // Add our custom JWT security filter
        http.addFilterBefore(jwtAuthenticationFilter(), UsernamePasswordAuthenticationFilter.class);
    }
}
```

```
package com.diploma.linguistic_glucose_analyzer.constants;

import com.diploma.linguistic_glucose_analyzer.dto.response.ApiResponse;
import com.diploma.linguistic_glucose_analyzer.model.Alphabet;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;

public interface LinguisticChainConstants {
    int MIN_GLUCOSE = 0;
    int MAX_GLUCOSE = 520;

    int GLUCOSE_MIN_HEALTHY = 40;
    int GLUCOSE_MAX_HEALTHY = 200;

    int MIN_MEASURES_PER_DAY_PER_PERSON = 2;

//    Alphabet USED_ALPHABET = Alphabet.ASCII100;
    Alphabet USED_ALPHABET = Alphabet.ENGLISH;

    String DEFAULT_ZONE = "+3";
```

```
//   String DEFAULT_ZONE = "America/Toronto";

   String DEFAULT_AUTHORITY = "USER";

   interface ResponseEntities {
      ResponseEntity BAD_REQ_USERNAME_TAKEN = new ResponseEntity<>(new ApiResponse(
            false, "Username is already taken!"
      ), HttpStatus.BAD_REQUEST);

      ResponseEntity BAD_REQ_EMAIL_TAKEN = new ResponseEntity<>(new ApiResponse(
            false, "Email Address already in use!"
      ), HttpStatus.BAD_REQUEST);

      ResponseEntity USER_REGISTERED_SUCCESSFULLY = new ResponseEntity<>(new ApiResponse(
            true, "User registered successfully"
      ), HttpStatus.CREATED);
   }
}
```

```
public interface PersonDAO extends
JpaRepository<Person, Long> {
}
```

```
package com.diploma.linguistic_glucose_analyzer.dao;

import
com.diploma.linguistic_glucose_analyzer.model.User;
import
org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface UserDAO extends JpaRepository<User,
Long>, ExtraUserDAO {
}
```

```
package com.diploma.linguistic_glucose_analyzer.dao;

import
com.diploma.linguistic_glucose_analyzer.dao.ExtraGlucos
eDAO;
import
com.diploma.linguistic_glucose_analyzer.model.GlucoseD
ataRecord;
import
org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface GlucoseDAO extends
JpaRepository<GlucoseDataRecord, Long>,
ExtraGlucoseDAO {
}
```

```
package com.diploma.linguistic_glucose_analyzer.dao;

import
com.diploma.linguistic_glucose_analyzer.model.Prediction
ChainOccasion;
import
org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface PredictionChainOccasionDAO
      extends JpaRepository<PredictionChainOccasion,
Long>, ExtraPredictionChainOccasionDAO {
}
```

```
package com.diploma.linguistic_glucose_analyzer.dao;

import
com.diploma.linguistic_glucose_analyzer.model.User;

import java.util.Optional;

public interface ExtraUserDAO {
   Optional<User> getUserByUsernameOrEmail(String
usernameOrEmail);
   boolean existsByUsername(String username);
   boolean existsByEmail(String email);
}
```

```
package com.diploma.linguistic_glucose_analyzer.dao;

import
com.diploma.linguistic_glucose_analyzer.model.Person;
import
org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
```

```
package com.diploma.linguistic_glucose_analyzer.dao;
```

```
public interface ExtraPredictionChainOccasionDAO {
}
```

```
package com.diploma.linguistic_glucose_analyzer.dao;

import
com.diploma.linguistic_glucose_analyzer.model.GlucoseD
ataRecord;
import org.springframework.stereotype.Repository;

import java.util.List;

public interface ExtraGlucoseDAO {
   List<GlucoseDataRecord> getRecordsByPerson(long
personId);
   List<GlucoseDataRecord> getRecordsByUser(long
userId);
}
```

```
package
com.diploma.linguistic_glucose_analyzer.dao.impl;

import
com.diploma.linguistic_glucose_analyzer.dao.ExtraUserD
AO;
import
com.diploma.linguistic_glucose_analyzer.model.User;
import org.springframework.stereotype.Repository;

import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import javax.persistence.TypedQuery;
import java.util.Optional;

@Repository
public class ExtraUserDAOImpl implements
ExtraUserDAO {

   @PersistenceContext
   private EntityManager entityManager;

   private static final String LOGIN_QUERY_PARAM =
"login";
   private static final String LOGIN_QUERY_PARAM_NAME
= ":" + LOGIN_QUERY_PARAM;
   private static final String EMAIL_QUERY_PARAM =
"email";
   private static final String EMAIL_QUERY_PARAM_NAME
= ":" + EMAIL_QUERY_PARAM;

   private static final String
GET_USER_BY_LOGIN_OF_EMAIL_QUERY =
         "FROM  User WHERE login = " +
LOGIN_QUERY_PARAM_NAME + " OR email = " +
EMAIL_QUERY_PARAM_NAME;

   private static final String GET_USER_BY_LOGIN_QUERY =
         "FROM  User WHERE login = " +
LOGIN_QUERY_PARAM_NAME;

   private static final String GET_USER_BY_EMAIL_QUERY =
         "FROM  User WHERE email = " +
EMAIL_QUERY_PARAM_NAME;


   @Override
   public Optional<User>
getUserByUsernameOrEmail(String usernameOrEmail) {
      TypedQuery<User> query =
entityManager.createQuery(GET_USER_BY_LOGIN_OF_EM
AIL_QUERY, User.class);

      query.setParameter(LOGIN_QUERY_PARAM,
usernameOrEmail);
      query.setParameter(EMAIL_QUERY_PARAM,
usernameOrEmail);

      try {
         return Optional.of(query.getSingleResult());
      } catch (Exception ex) {
         return Optional.empty();
      }
   }

   @Override
   public boolean existsByUsername(String username) {
      TypedQuery<User> query =
entityManager.createQuery(GET_USER_BY_LOGIN_QUERY,
User.class);

      query.setParameter(LOGIN_QUERY_PARAM,
username);

      return !query.getResultList().isEmpty();
   }

   @Override
   public boolean existsByEmail(String email) {
      TypedQuery<User> query =
entityManager.createQuery(GET_USER_BY_EMAIL_QUERY,
User.class);

      query.setParameter(EMAIL_QUERY_PARAM, email);

      return !query.getResultList().isEmpty();
   }
}
```

```
package
com.diploma.linguistic_glucose_analyzer.dao.impl;

import
com.diploma.linguistic_glucose_analyzer.dao.ExtraGlucos
eDAO;
import
com.diploma.linguistic_glucose_analyzer.model.GlucoseD
ataRecord;
import lombok.extern.slf4j.Slf4j;
import org.springframework.context.annotation.Primary;
import org.springframework.stereotype.Repository;

import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
```

```
import javax.persistence.TypedQuery;
import java.util.List;

@Repository
@Primary
@Slf4j
public class ExtraGlucoseDAOImpl implements
ExtraGlucoseDAO {

   @PersistenceContext
   private EntityManager entityManager;

   private static final String PERSON_ID_QUERY_PARAM =
"personId";
   private static final String
PERSON_ID_QUERY_PARAM_NAME = ":personId";

   private static final String
GET_GLUCOSE_RECORDS_BY_PERSON_ID_QUERY =
         "FROM GlucoseDataRecord WHERE person.id = " +
PERSON_ID_QUERY_PARAM_NAME;

   private static final String USER_ID_QUERY_PARAM =
"userId";
   private static final String
USER_ID_QUERY_PARAM_NAME = ":userId";

   private static final String
GET_GLUCOSE_RECORDS_BY_USER_ID_QUERY =
         "FROM GlucoseDataRecord WHERE person.id =
(SELECT person.id from User WHERE id = " +
USER_ID_QUERY_PARAM_NAME + ")";


   @Override
   public List<GlucoseDataRecord>
getRecordsByPerson(long personId) {
      TypedQuery<GlucoseDataRecord> query =
entityManager.createQuery(GET_GLUCOSE_RECORDS_BY_
PERSON_ID_QUERY, GlucoseDataRecord.class);

      query.setParameter(PERSON_ID_QUERY_PARAM,
personId);

      return query.getResultList();
   }


   @Override
   public List<GlucoseDataRecord>
getRecordsByUser(long userId) {
      TypedQuery<GlucoseDataRecord> query =
entityManager.createQuery(GET_GLUCOSE_RECORDS_BY_
USER_ID_QUERY, GlucoseDataRecord.class);

      query.setParameter(USER_ID_QUERY_PARAM,
userId);

      return query.getResultList();
   }
}


package com.diploma.linguistic_glucose_analyzer.dto;

import lombok.AllArgsConstructor;
import lombok.Data;

import java.time.Instant;

@Data
@AllArgsConstructor
public class PossibleFutureGlucoseDTO {
   private Instant approximateTime;
   private Double min;
   private Double max;
}



package com.diploma.linguistic_glucose_analyzer.dto;

import lombok.AllArgsConstructor;
import lombok.Data;

@Data
@AllArgsConstructor
public class SymbolBounds {
   private char symbol;
   private double min;
   private double max;
}



package com.diploma.linguistic_glucose_analyzer.dto;

import lombok.Data;

import java.time.Instant;

@Data
public class UserSummaryDTO {
   private String username;
   private String email;
   private String name;
   private String surname;
   private Instant birthDate;
   private long diabetesTypeId;
}



package
com.diploma.linguistic_glucose_analyzer.dto.request;

import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

import javax.validation.constraints.NotBlank;

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
public class LoginRequest {
```

```
    @NotBlank
    private String usernameOrEmail;

    @NotBlank
    private String password;
}
```

```
package
com.diploma.linguistic_glucose_analyzer.dto.request;

import lombok.Getter;
import lombok.Setter;

import javax.validation.constraints.*;
import java.time.Instant;

@Getter
@Setter
public class SignUpRequest {

    @NotBlank
    @Size(min = 4, max = 64)
    private String username;

    @Size(max = 32)
    private String firstName;

    @Size(max = 64)
    private String lastName;

    @NotBlank
    @Size(max = 128)
    @Email
    private String email;

    @NotBlank
    @Size(min = 6, max = 128)
    private String password;

    @NotNull
    @Min(1)
    @Max(3)
    private Long diabetesTypeId;

    private Instant birthDate;

    private String role;
}
```

```
package
com.diploma.linguistic_glucose_analyzer.dto.response;

import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.Setter;

@Getter
@Setter
@AllArgsConstructor
public class ApiResponse {

    private Boolean success;
    private String message;
}
```

```
package
com.diploma.linguistic_glucose_analyzer.dto.response;

import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

import java.util.Map;

@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor
public class DashboardResponse {

    private Map<Integer, Long> usersPerYear;
    private Map<String, Long> servicesDistribution;
}
```

```
package
com.diploma.linguistic_glucose_analyzer.dto.response;

import lombok.Getter;
import lombok.Setter;

@Getter
@Setter
public class JwtAuthenticationResponse {

    private String accessToken;
    private String tokenType = "Bearer";

    public JwtAuthenticationResponse(String accessToken)
{
        this.accessToken = accessToken;
    }
}
```

```
package
com.diploma.linguistic_glucose_analyzer.exception;

import org.springframework.http.HttpStatus;
import
org.springframework.web.bind.annotation.ResponseStatu
s;

@ResponseStatus(HttpStatus.INTERNAL_SERVER_ERROR)
public class AppException extends RuntimeException {
```

```
  public AppException(String message) {
    super(message);
  }

  public AppException(String message, Throwable cause)
{
    super(message, cause);
  }
}
```

```
package
com.diploma.linguistic_glucose_analyzer.exception;

import org.springframework.http.HttpStatus;
import
org.springframework.web.bind.annotation.ResponseStatu
s;

@ResponseStatus(HttpStatus.BAD_REQUEST)
public class BadRequestException extends
RuntimeException {

  public BadRequestException(String message) {
    super(message);
  }

  public BadRequestException(String message,
Throwable cause) {
    super(message, cause);
  }
}
```

```
package
com.diploma.linguistic_glucose_analyzer.exception;

import lombok.Getter;
import org.springframework.http.HttpStatus;
import
org.springframework.web.bind.annotation.ResponseStatu
s;

@Getter
@ResponseStatus(HttpStatus.NOT_FOUND)
public class ResourceNotFoundException extends
RuntimeException {

  private String resourceName;
  private String fieldName;
  private Object fieldValue;

  public ResourceNotFoundException(String
resourceName, String fieldName, Object fieldValue) {
    super(String.format("%s not found with %s : '%s'",
resourceName, fieldName, fieldValue));
    this.resourceName = resourceName;
    this.fieldName = fieldName;
    this.fieldValue = fieldValue;
  }
}
```

```
package
com.diploma.linguistic_glucose_analyzer.service.impl;

import
com.diploma.linguistic_glucose_analyzer.service.CrudServ
ice;
import
org.springframework.data.jpa.repository.JpaRepository;

import java.util.List;

public class AbstractCRUDService<T, I> implements
CrudService<T, I> {
  private final JpaRepository<T, I> repository;

  public AbstractCRUDService(JpaRepository<T, I>
repository) {
    this.repository = repository;
  }

  @Override
  public T getById(I id) {
    return repository.findById(id).orElse(null);
  }

  @Override
  public List<T> getAll() {
    return repository.findAll();
  }

  @Override
  public T save(T entity) {
    return repository.save(entity);
  }

  @Override
  public List<T> saveAll(List<T> entities) {
    return repository.saveAll(entities);
  }

  @Override
  public void delete(T entity) {
    repository.delete(entity);
  }

  @Override
  public void deleteById(I id) {
    repository.deleteById(id);
  }
}
```

```
package
com.diploma.linguistic_glucose_analyzer.service.impl;

import
com.diploma.linguistic_glucose_analyzer.dto.SymbolBoun
ds;
import
com.diploma.linguistic_glucose_analyzer.model.Alphabet;
import
com.diploma.linguistic_glucose_analyzer.model.GlucoseD
```

```
ataRecord;
import
com.diploma.linguistic_glucose_analyzer.service.LinguisticChainService;

import java.util.Arrays;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Map;
import java.util.function.Function;

import static
com.diploma.linguistic_glucose_analyzer.constants.LinguisticChainConstants.*;

public abstract class AbstractLinguisticChainService
implements LinguisticChainService {

  /**
   * Distribution function used to determine the distribution
of numbers between symbols
   * @return
   */
  protected abstract Function<Double, Double>
getDistributionFunc();

  /**
   * Where to start the distribution from distribution
function
   * @return
   */
  protected abstract double getStartFrom();

  /**
   * Where to end the distribution from distribution
function
   * @return
   */
  protected abstract double getFinishAt();

  private Map<Character, Double>
getPerSymbolDistribution(Alphabet alphabet) {
    char[] symbols = alphabet.getSymbols();
    double step = (getFinishAt() - getStartFrom()) /
symbols.length;

    Map<Character, Double> funcValues = new
LinkedHashMap<>();
    double sum = 0;

    for (int i = 0; i < symbols.length; i++) {
      double value =
getDistributionFunc().apply(getStartFrom() + step * i);
      sum += value;
      funcValues.put(symbols[i], value);
    }

    Map<Character, Double> perSymbolDistribution =
new LinkedHashMap<>();

    for (char symbol : symbols) {
      perSymbolDistribution.put(symbol,
funcValues.get(symbol) / sum);
    }

    return perSymbolDistribution;
  }

  //TODO Make caching for used_alphabet
  private Map<Character, Double>
getUpperSymbolBounds(Alphabet alphabet) {
    Map<Character, Double> upperSymbolBounds = new
LinkedHashMap<>();

    char[] symbols = alphabet.getSymbols();
    Map<Character, Double> perSymbolDistribution =
getPerSymbolDistribution(alphabet);
    double currUpperBound = 0;

    for (char symbol : symbols) {
      currUpperBound +=
perSymbolDistribution.get(symbol) * (MAX_GLUCOSE -
MIN_GLUCOSE);
      upperSymbolBounds.put(symbol,
currUpperBound);
    }

    return upperSymbolBounds;
  }

  @Override
  public String getChain(List<GlucoseDataRecord>
records) {
    return getChain(records, USED_ALPHABET);
  }


  @Override
  public String getChain(List<GlucoseDataRecord>
records, Alphabet alphabet) {
    Map<Character, Double> upperSymbolBounds =
getUpperSymbolBounds(alphabet);
    StringBuilder linguisticChain = new StringBuilder();
    for (GlucoseDataRecord record : records) {

linguisticChain.append(getSymbol(record.getValue(),
upperSymbolBounds, alphabet));
    }

    return linguisticChain.toString();
  }

  private char getSymbol(double value, Map<Character,
Double> upperSymbolBounds, Alphabet alphabet) {
    if (value < MIN_GLUCOSE || value > MAX_GLUCOSE) {
      throw new IllegalArgumentException();
    }

    for (Map.Entry<Character, Double>
upperSymbolBound : upperSymbolBounds.entrySet()) {
      if (value < upperSymbolBound.getValue()) {
        return upperSymbolBound.getKey();
      }
    }

    return
alphabet.getSymbols()[alphabet.getSymbols().length - 1];
  }

  @Override
  public boolean isHyperglycemic(char symbol) {
    return isHyperglycemic(symbol, USED_ALPHABET);
  }
```

```
    @Override
    public boolean isHyperglycemic(char symbol, Alphabet
alphabet) {
        char[] symbols = alphabet.getSymbols();

        if (symbol == symbols[0]) {
            return false;
        }

        return
getUpperSymbolBounds(alphabet).get(symbol) >
GLUCOSE_MAX_HEALTHY;
    }

    @Override
    public boolean isHypoglycemic(char symbol) {
        return isHypoglycemic(symbol, USED_ALPHABET);
    }

    @Override
    public boolean isHypoglycemic(char symbol, Alphabet
alphabet) {
        char[] symbols = alphabet.getSymbols();

        if (symbol == symbols[0]) {
            return true;
        }

        int index =  Arrays.binarySearch(symbols, symbol);

        if (index == -1) {
            return false;
        }

        return
getUpperSymbolBounds(alphabet).get(symbols[index -
1]) < GLUCOSE_MIN_HEALTHY;
    }

    @Override
    public SymbolBounds getSymbolBounds(char symbol) {
        return getSymbolBounds(symbol, USED_ALPHABET);
    }

    @Override
    public SymbolBounds getSymbolBounds(char symbol,
Alphabet alphabet) {
        char[] symbols = alphabet.getSymbols();

        Map<Character, Double> upperBounds =
getUpperSymbolBounds(alphabet);

        if (symbol == symbols[0]) {
            return new SymbolBounds(symbol, MIN_GLUCOSE,
upperBounds.get(symbol));
        }

        int index =  Arrays.binarySearch(symbols, symbol);

        if (index == -1) {
            return null;
        }

        return new SymbolBounds(symbol,
upperBounds.get(symbols[index - 1]),
upperBounds.get(symbol));
    }

    //TODO Remove

    @Override
    public Map<Character, Double>
getSymbolsUpperBound() {
        return getUpperSymbolBounds(USED_ALPHABET);
    }
}
```

```
package
com.diploma.linguistic_glucose_analyzer.service.impl;

import
com.diploma.linguistic_glucose_analyzer.auth.JwtTokenPr
ovider;
import
com.diploma.linguistic_glucose_analyzer.constants.Lingui
sticChainConstants;
import
com.diploma.linguistic_glucose_analyzer.dao.PersonDAO;
import
com.diploma.linguistic_glucose_analyzer.dao.UserDAO;
import
com.diploma.linguistic_glucose_analyzer.dto.request.Logi
nRequest;
import
com.diploma.linguistic_glucose_analyzer.dto.request.Sign
UpRequest;
import
com.diploma.linguistic_glucose_analyzer.model.DiabetesT
ype;
import
com.diploma.linguistic_glucose_analyzer.model.Person;
import
com.diploma.linguistic_glucose_analyzer.model.User;
import
com.diploma.linguistic_glucose_analyzer.service.AuthServ
ice;
import lombok.extern.slf4j.Slf4j;
import
org.springframework.beans.factory.annotation.Autowired
;
import org.springframework.http.ResponseEntity;
import
org.springframework.security.authentication.Authenticati
onManager;
import
org.springframework.security.authentication.UsernameP
asswordAuthenticationToken;
import
org.springframework.security.core.Authentication;
import
org.springframework.security.core.context.SecurityConte
xtHolder;
import
org.springframework.security.crypto.password.Password
Encoder;
import org.springframework.stereotype.Service;
import
org.springframework.transaction.annotation.Transaction
al;
```

```
import java.sql.Timestamp;
import java.time.Instant;

@Slf4j
@Service
public class AuthServiceImpl implements AuthService {

  @Autowired
  private UserDAO userDao;

  @Autowired
  private PersonDAO personDAO;

  @Transactional
  public ResponseEntity registerUser(SignUpRequest
request, PasswordEncoder encoder) {

    if
(userDao.existsByUsername(request.getUsername())) {
      return
LinguisticChainConstants.ResponseEntities.BAD_REQ_USE
RNAME_TAKEN;
    }
    if (userDao.existsByEmail(request.getEmail())) {
      return
LinguisticChainConstants.ResponseEntities.BAD_REQ_EM
AIL_TAKEN;
    }

    Person person =
        new Person(request.getDiabetesTypeId(),
request.getFirstName(), request.getLastName(),
request.getBirthDate());

    personDAO.save(person);

    User user = new User(person, request.getUsername(),
        encoder.encode(request.getPassword()),
request.getEmail(), Timestamp.from(Instant.now()));

    userDao.save(user);

    return
LinguisticChainConstants.ResponseEntities.USER_REGIST
ERED_SUCCESSFULLY;
  }

  public String authenticateUser(LoginRequest request,
AuthenticationManager authManager, JwtTokenProvider
tokenProvider) {
    log.debug("Request: {}|{}",
request.getUsernameOrEmail(), request.getPassword());
    Authentication authentication =
authManager.authenticate(
        new UsernamePasswordAuthenticationToken(
            request.getUsernameOrEmail(),
            request.getPassword()
        )
    );


SecurityContextHolder.getContext().setAuthentication(au
thentication);

    return
tokenProvider.generateToken(authentication);
  }
}


package
com.diploma.linguistic_glucose_analyzer.service.impl;

import
com.diploma.linguistic_glucose_analyzer.constants.Lingui
sticChainConstants;
import
com.diploma.linguistic_glucose_analyzer.service.GlucoseF
ileService;
import
com.diploma.linguistic_glucose_analyzer.model.GlucoseD
ataCode;
import
com.diploma.linguistic_glucose_analyzer.model.GlucoseD
ataRecord;
import lombok.extern.slf4j.Slf4j;
import org.springframework.context.annotation.Primary;
import org.springframework.stereotype.Repository;

import java.io.File;
import java.io.IOException;
import java.nio.file.Files;
import java.time.Instant;
import java.time.LocalDateTime;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.DateTimeParseException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.Locale;
import java.util.stream.Stream;

@Slf4j
@Primary
@Repository
public class ContinuousGlucoseFileServiceImpl
implements GlucoseFileService {

  /**
   * Fetch all records from file
   *
   * @param filePath relative path to file
   * @return list of all records from file
   */
  @Override
  public List<GlucoseDataRecord> getRecords(String
filePath) {
    log.info("Reading from file: {}", filePath);

    if (filePath == null) {
      log.debug("Received null as param");
      return null;
    }

    var recordsFile = new File(filePath);

    return getRecords(recordsFile);
  }

  @Override
  public List<GlucoseDataRecord> getRecords(File file) {
```

```
        if (file.exists()) {
            var records = new
ArrayList<GlucoseDataRecord>();
            try (Stream<String> linesStream =
Files.lines(file.toPath())) {
                linesStream.forEach(line -> {
                    String[] recordData = line.split(",", -1);
                    GlucoseDataRecord record =
getRecord(recordData);
                    if (record != null) {
                        records.add(record);
                    }
                });
            } catch (IOException ex) {
                log.debug("Error while reading file: ", ex);
            }

            log.trace("Fetched records: {}", records);
            return records;
        }

        log.debug("No file present");
        return null;
    }

    @Override
    public List<GlucoseDataRecord> getRecords(String[]
rows) {
        var records = new ArrayList<GlucoseDataRecord>();

        for (String line : rows) {
            String[] recordData = line.split(",", -1);
            GlucoseDataRecord record =
getRecord(recordData);
            if (record != null) {
                records.add(record);
            }
        }

        log.trace("Fetched records: {}", records);
        return records;
    }

    private GlucoseDataRecord getRecord(String[]
recordData) {
        if (recordData == null || recordData.length < 3) {
            log.debug("Invalid data received: {}",
Arrays.toString(recordData));
            return null;
        }

        var timeToParse = recordData[1] + ", " +
recordData[0];

        Instant eventTime;
        try {
            eventTime = LocalDateTime.parse(timeToParse,
DateTimeFormatter.ofPattern("H:mm:ss',' uuuu-M-d",
Locale.US))

.atZone(ZoneId.of(LinguisticChainConstants.DEFAULT_ZO
NE))
                    .toInstant();
        } catch (DateTimeParseException ex) {
            log.debug("Error while parsing date: {}",
timeToParse);
            log.debug("Stacktrace: ", ex);
            return null;
        }

        double value;

        try {
            value = Double.parseDouble(recordData[2]);
        } catch (NumberFormatException ex) {
            log.debug("Error while parsing value({})",
recordData[2]);
            log.debug("Stacktrace: ", ex);
            return null;
        }

        return new GlucoseDataRecord(eventTime,
GlucoseDataCode.UNSPECIFIED_BLOOD_GLUCOSE_MEASU
REMENT, (int) Math.floor(value * 18));
    }
}
```

```
package
com.diploma.linguistic_glucose_analyzer.service.impl;

import
com.diploma.linguistic_glucose_analyzer.dao.GlucoseDAO;
import
com.diploma.linguistic_glucose_analyzer.model.GlucoseD
ataCode;
import
com.diploma.linguistic_glucose_analyzer.model.GlucoseD
ataRecord;
import
com.diploma.linguistic_glucose_analyzer.service.GlucoseS
ervice;
import lombok.extern.slf4j.Slf4j;
import
org.springframework.beans.factory.annotation.Autowired
;
import org.springframework.stereotype.Service;

import java.util.List;

@Slf4j
@Service
public class GlucoseServiceImpl extends
AbstractCRUDService<GlucoseDataRecord, Long>
        implements GlucoseService {

    private GlucoseDAO glucoseDAO;

    @Autowired
    public GlucoseServiceImpl(GlucoseDAO glucoseDAO) {
        super(glucoseDAO);
        this.glucoseDAO = glucoseDAO;
    }

    @Override
    public List<GlucoseDataRecord>
getRecordsByPerson(long personId) {
        return glucoseDAO.getRecordsByPerson(personId);
    }

    @Override
```

```
    public List<GlucoseDataRecord>
getRecordsByUser(long userId) {
        return glucoseDAO.getRecordsByUser(userId);
    }

    @Override
    public GlucoseDataRecord save(GlucoseDataRecord
entity) {

entity.setCode(GlucoseDataCode.UNSPECIFIED_BLOOD_G
LUCOSE_MEASUREMENT);
        return super.save(entity);
    }
}
```

```
package
com.diploma.linguistic_glucose_analyzer.service.impl;

import
com.diploma.linguistic_glucose_analyzer.model.Alphabet;
import
com.diploma.linguistic_glucose_analyzer.model.GlucoseD
ataRecord;
import
com.diploma.linguistic_glucose_analyzer.service.Linguisti
cChainService;
import lombok.extern.slf4j.Slf4j;
import org.springframework.context.annotation.Primary;
import org.springframework.stereotype.Service;

import java.util.Arrays;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Map;
import java.util.function.Function;

import static
com.diploma.linguistic_glucose_analyzer.constants.Lingui
sticChainConstants.*;

@Slf4j
//@Primary
@Service
public class LogNormalLinguisticChainService extends
AbstractLinguisticChainService implements
LinguisticChainService {

    private final static double o = 0.5;
    private final static double u = 0;

    private final static double START_DISTRIBUTION_FROM
= 0.7;
    private final static double FINISH_DISTRIBUTION_AT =
3.3;

    /**
     * Distribution function used to determine the distribution
of numbers between symbols
     */
    @Override
    protected Function<Double, Double>
getDistributionFunc() {
        return x -> (1 / x * o * Math.sqrt(2 * Math.PI)) *
Math.exp(-Math.pow((Math.log(x) - u), 2) / (2 * o * o));
    }

    /**
     * Where to start the distribution from distribution
function
     */
    @Override
    protected double getStartFrom() {
        return START_DISTRIBUTION_FROM;
    }

    /**
     * Where to end the distribution from distribution
function
     */
    @Override
    protected double getFinishAt() {
        return FINISH_DISTRIBUTION_AT;
    }
}
```

```
package
com.diploma.linguistic_glucose_analyzer.service.impl;

import
com.diploma.linguistic_glucose_analyzer.service.Linguisti
cChainService;
import lombok.extern.slf4j.Slf4j;
import org.springframework.context.annotation.Primary;
import org.springframework.stereotype.Service;

import java.util.function.Function;

@Slf4j
@Primary
@Service
public class ParabolicLinguisticChainService extends
AbstractLinguisticChainService implements
LinguisticChainService {

//    private final static Double POWER = 2.0;
//    private final static Double FOLD = 1.0;
//
//    private final static Double X_OFFSET = 1.0;
//    private final static Double Y_OFFSET = 0.3;
//
//    private final static Double START_FROM = 0.0;
//    private final static Double FINISH_AT = 2.5;

    private final static Double POWER = 4.0;
    private final static Double FOLD = 5.0;

    private final static Double X_OFFSET = 0.7;
    private final static Double Y_OFFSET = 0.3;

    private final static Double START_FROM = 0.0;
    private final static Double FINISH_AT = 1.7;

    @Override
    protected Function<Double, Double>
getDistributionFunc() {
        return x -> FOLD * Math.pow(x - X_OFFSET, POWER) +
```

```
Y_OFFSET;
  }

  @Override
  protected double getStartFrom() {
    return START_FROM;
  }

  @Override
  protected double getFinishAt() {
    return FINISH_AT;
  }
}
```

```
package
com.diploma.linguistic_glucose_analyzer.service.impl;

import
com.diploma.linguistic_glucose_analyzer.dao.PersonDAO;
import
com.diploma.linguistic_glucose_analyzer.model.Person;
import
com.diploma.linguistic_glucose_analyzer.service.PersonSe
rvice;
import
org.springframework.beans.factory.annotation.Autowired
;
import org.springframework.stereotype.Service;

@Service
public class PersonServiceImpl extends
AbstractCRUDService<Person, Long>
    implements PersonService {
  private PersonDAO personDAO;

  @Autowired
  public PersonServiceImpl(PersonDAO personDAO) {
    super(personDAO);
    this.personDAO = personDAO;
  }
}
```

```
package
com.diploma.linguistic_glucose_analyzer.service.impl;

import
com.diploma.linguistic_glucose_analyzer.dto.PossibleFutu
reGlucoseDTO;
import
com.diploma.linguistic_glucose_analyzer.dto.SymbolBoun
ds;
import
com.diploma.linguistic_glucose_analyzer.model.GlucoseD
ataRecord;
import
com.diploma.linguistic_glucose_analyzer.service.GlucoseS
ervice;
import
com.diploma.linguistic_glucose_analyzer.service.Linguisti
cChainService;
import
com.diploma.linguistic_glucose_analyzer.service.Predictio
nService;
import
com.diploma.linguistic_glucose_analyzer.service.matrix.Pr
edictionMatrixFactory;
import
com.diploma.linguistic_glucose_analyzer.service.matrix.m
odel.PredictionMatrix;
import lombok.extern.slf4j.Slf4j;
import
org.springframework.beans.factory.annotation.Autowired
;
import org.springframework.stereotype.Service;

import java.time.Instant;
import java.util.List;
import java.util.Map;

@Slf4j
@Service
public class PredictionServiceImpl implements
PredictionService {

  private static final int CHAIN_LENGTH = 1;
  private static final int RECORDS_TO_PREDICT = 5;

  private GlucoseService glucoseService;
  private LinguisticChainService linguisticChainService;
  private PredictionMatrixFactory
predictionMatrixFactory;

  @Autowired
  public PredictionServiceImpl(GlucoseService
glucoseService, LinguisticChainService
linguisticChainService, PredictionMatrixFactory
predictionMatrixFactory) {
    this.glucoseService = glucoseService;
    this.linguisticChainService = linguisticChainService;
    this.predictionMatrixFactory =
predictionMatrixFactory;
  }

  @Override
  public PossibleFutureGlucoseDTO
getPossibleFutureGlucoseValueForUser(long userId) {
    List<GlucoseDataRecord> allRecords =
glucoseService.getAll();
    PredictionMatrix predictionMatrix =
predictionMatrixFactory.getMatrix(allRecords,
CHAIN_LENGTH);

    List<GlucoseDataRecord> userRecords =
glucoseService.getRecordsByUser(userId);
    String userRecordsChain =
linguisticChainService.getChain(userRecords);
    char lastSymbol =
userRecordsChain.charAt(userRecordsChain.length() - 1);

    for (int i = 0; i < RECORDS_TO_PREDICT; i++) {
      Map<Character, Double> appearanceChances =
predictionMatrix.getAppearanceChances().get(String.valu
eOf(lastSymbol));

      if (appearanceChances == null) {
        log.debug("UNEXPECTED LAST SYMBOL [{}].
Terminating early", lastSymbol);
        break;
      }
```

```
        double currMax = Double.MIN_VALUE;
        for (Map.Entry<Character, Double>
appearanceChance : appearanceChances.entrySet()) {
            if (appearanceChance.getValue() > currMax) {
                currMax = appearanceChance.getValue();
                lastSymbol = appearanceChance.getKey();
            }
        }
    }

    Instant lastGlucoseRecordTime =
userRecords.get(userRecords.size() - 1).getEventTime();
    SymbolBounds symbolBounds =
linguisticChainService.getSymbolBounds(lastSymbol);

    return new
PossibleFutureGlucoseDTO(lastGlucoseRecordTime,
symbolBounds.getMin(), symbolBounds.getMax());
  }
}
```

```
package
com.diploma.linguistic_glucose_analyzer.service.impl;

import
com.diploma.linguistic_glucose_analyzer.constants.Linguis
ticChainConstants;
import
com.diploma.linguistic_glucose_analyzer.model.Alphabet;
import
com.diploma.linguistic_glucose_analyzer.model.GlucoseD
ataRecord;
import
com.diploma.linguistic_glucose_analyzer.service.Linguisti
cChainService;
import
com.diploma.linguistic_glucose_analyzer.service.impl.Abst
ractLinguisticChainService;
import
com.diploma.linguistic_glucose_analyzer.utils.GlucoseDat
aCodeUtils;
import lombok.extern.slf4j.Slf4j;
import org.springframework.context.annotation.Primary;
import org.springframework.stereotype.Service;

import java.util.Arrays;
import java.util.List;
import java.util.function.Function;

import static
com.diploma.linguistic_glucose_analyzer.constants.Linguis
ticChainConstants.*;

@Slf4j
//@Primary
@Service
public class UniformLinguisticChainService extends
AbstractLinguisticChainService implements
LinguisticChainService {

  @Override
  protected Function<Double, Double>
getDistributionFunc() {
    return x -> 1.0;
  }

  @Override
  protected double getStartFrom() {
    return 0;
  }

  @Override
  protected double getFinishAt() {
    return 100;
  }
}
```

```
package
com.diploma.linguistic_glucose_analyzer.service.impl;

import
com.diploma.linguistic_glucose_analyzer.dao.UserDAO;
import
com.diploma.linguistic_glucose_analyzer.dto.UserSummar
yDTO;
import
com.diploma.linguistic_glucose_analyzer.exception.AppEx
ception;
import
com.diploma.linguistic_glucose_analyzer.model.Person;
import
com.diploma.linguistic_glucose_analyzer.model.User;
import
com.diploma.linguistic_glucose_analyzer.service.PersonSe
rvice;
import
com.diploma.linguistic_glucose_analyzer.service.UserServ
ice;
import
org.springframework.beans.factory.annotation.Autowired
;
import org.springframework.stereotype.Service;

import javax.transaction.Transactional;

@Service
public class UserServiceImpl extends
AbstractCRUDService<User, Long>
    implements UserService {
  private UserDAO userDAO;

  private PersonService personService;

  @Autowired
  public UserServiceImpl(UserDAO userDAO,
PersonService personService) {
    super(userDAO);
    this.userDAO = userDAO;
    this.personService = personService;
  }

  @Override
  public UserSummaryDTO getUserSummary(long
userId) {
    User user = userDAO.findById(userId)
```

```
            .orElseThrow(() -> new AppException("No user
with id" + userId + "was found"));

        Person userPersonalData = user.getPerson();

        UserSummaryDTO summary = new
UserSummaryDTO();

        summary.setUsername(user.getLogin());
        summary.setEmail(user.getEmail());
        summary.setName(userPersonalData.getName());

summary.setSurname(userPersonalData.getSurname());

summary.setBirthDate(userPersonalData.getBirthDate());

summary.setDiabetesTypeId(userPersonalData.getDiabet
esTypeId());

        return summary;
    }

    @Override
    @Transactional
    public void updateUserData(long userId,
UserSummaryDTO summary) {
        User user = userDAO.findById(userId)
            .orElseThrow(() -> new AppException("No user
with id" + userId + "was found"));

        Person userPersonalData = user.getPerson();

        user.setLogin(summary.getUsername());
        user.setEmail(summary.getEmail());
        userPersonalData.setName(summary.getName());

userPersonalData.setSurname(summary.getSurname());

userPersonalData.setBirthDate(summary.getBirthDate());

userPersonalData.setDiabetesTypeId(summary.getDiabet
esTypeId());

        userDAO.save(user);
        personService.save(userPersonalData);
    }
}
```

```
package com.diploma.linguistic_glucose_analyzer.service;

import
com.diploma.linguistic_glucose_analyzer.auth.JwtTokenPr
ovider;
import
com.diploma.linguistic_glucose_analyzer.dto.request.Logi
nRequest;
import
com.diploma.linguistic_glucose_analyzer.dto.request.Sign
UpRequest;
import org.springframework.http.ResponseEntity;
import
org.springframework.security.authentication.Authenticati
onManager;
import
org.springframework.security.crypto.password.Password
Encoder;

public interface AuthService {
    ResponseEntity registerUser(SignUpRequest request,
PasswordEncoder encoder);
    String authenticateUser(LoginRequest request,
AuthenticationManager authManager, JwtTokenProvider
tokenProvider);
}
```

```
package com.diploma.linguistic_glucose_analyzer.service;

import java.util.List;

public interface CrudService<T, I> {
    T getById(I id);

    List<T> getAll();

    T save(T entity);

    List<T> saveAll(List<T> entities);

    void delete(T entity);

    void deleteById(I id);
}
```

```
package com.diploma.linguistic_glucose_analyzer.service;

import
com.diploma.linguistic_glucose_analyzer.model.GlucoseD
ataRecord;
import org.springframework.stereotype.Repository;

import java.io.File;
import java.util.List;

public interface GlucoseFileService {
    List<GlucoseDataRecord> getRecords(String filePath);
    List<GlucoseDataRecord> getRecords(File file);
    List<GlucoseDataRecord> getRecords(String[] rows);
}
```

```
package com.diploma.linguistic_glucose_analyzer.service;

import
com.diploma.linguistic_glucose_analyzer.model.GlucoseD
ataRecord;
import org.springframework.stereotype.Service;

import java.util.List;

public interface GlucoseService extends
```

```
CrudService<GlucoseDataRecord, Long> {
   List<GlucoseDataRecord> getRecordsByPerson(long
personId);
   List<GlucoseDataRecord> getRecordsByUser(long
userId);
}
```

```
package com.diploma.linguistic_glucose_analyzer.service;

import
com.diploma.linguistic_glucose_analyzer.dto.SymbolBoun
ds;
import
com.diploma.linguistic_glucose_analyzer.model.Alphabet;
import
com.diploma.linguistic_glucose_analyzer.model.GlucoseD
ataRecord;
import
com.diploma.linguistic_glucose_analyzer.service.filter.Rec
ordFilter;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.Map;

public interface LinguisticChainService {
   String getChain(List<GlucoseDataRecord> records);

   String getChain(List<GlucoseDataRecord> records,
Alphabet alphabet);

   boolean isHypoglycemic(char symbol);

   boolean isHypoglycemic(char symbol, Alphabet
alphabet);

   boolean isHyperglycemic(char symbol);

   boolean isHyperglycemic(char symbol, Alphabet
alphabet);

   SymbolBounds getSymbolBounds(char symbol);

   SymbolBounds getSymbolBounds(char symbol,
Alphabet alphabet);

   //TODO Remove
   Map<Character, Double> getSymbolsUpperBound();
}
```

```
package com.diploma.linguistic_glucose_analyzer.service;

import
com.diploma.linguistic_glucose_analyzer.model.Person;

public interface PersonService extends
CrudService<Person, Long> {
}
```

```
package com.diploma.linguistic_glucose_analyzer.service;

import
com.diploma.linguistic_glucose_analyzer.dto.PossibleFutu
reGlucoseDTO;

public interface PredictionService {
   PossibleFutureGlucoseDTO
getPossibleFutureGlucoseValueForUser(long userId);
}
```

```
package com.diploma.linguistic_glucose_analyzer.service;

import
com.diploma.linguistic_glucose_analyzer.dto.UserSummar
yDTO;
import
com.diploma.linguistic_glucose_analyzer.model.User;

public interface UserService extends CrudService<User,
Long>  {
   UserSummaryDTO getUserSummary(long userId);
   void updateUserData(long userId, UserSummaryDTO
summary);
}
```

```
package
com.diploma.linguistic_glucose_analyzer.service.filter;

import
com.diploma.linguistic_glucose_analyzer.model.GlucoseD
ataRecord;

import java.util.List;

public interface RecordFilter {
   List<GlucoseDataRecord>
filter(List<GlucoseDataRecord> records);
}
```

```
package
com.diploma.linguistic_glucose_analyzer.service.filter;

import
com.diploma.linguistic_glucose_analyzer.constants.Linguis
ticChainConstants;
import
com.diploma.linguistic_glucose_analyzer.model.GlucoseD
ataRecord;

import java.time.ZoneId;
import java.time.ZonedDateTime;
import java.time.temporal.ChronoUnit;
import java.util.ArrayList;
```

```
import java.util.Collections;
import java.util.List;
import java.util.stream.Collectors;

public class MinMeasuresPerDayRecordFilter implements
RecordFilter {
   private int minMeasures;

   public MinMeasuresPerDayRecordFilter(int
minMeasures) {
      this.minMeasures = minMeasures;
   }

   @Override
   public List<GlucoseDataRecord>
filter(List<GlucoseDataRecord> records) {
      List<ZonedDateTime> eventTimes = records.stream()
            .map(GlucoseDataRecord::getEventTime)
            .map(eventTime -> eventTime

.atZone(ZoneId.of(LinguisticChainConstants.DEFAULT_ZO
NE))
                  .truncatedTo(ChronoUnit.DAYS)
            )
            .collect(Collectors.toList());

      return eventTimes.stream()
            .filter(i -> Collections.frequency(eventTimes, i) >=
minMeasures)
            .count() == records.size()
            ? records
            : new ArrayList<>();
   }
}
```

```
package
com.diploma.linguistic_glucose_analyzer.service.filter;

import
com.diploma.linguistic_glucose_analyzer.model.GlucoseD
ataRecord;
import
com.diploma.linguistic_glucose_analyzer.utils.GlucoseDat
aCodeUtils;

import java.util.List;
import java.util.stream.Collectors;

public class GlucoseOnlyRecordFilter implements
RecordFilter {
   @Override
   public List<GlucoseDataRecord>
filter(List<GlucoseDataRecord> records) {
      return records.stream()
            .filter(rec ->
GlucoseDataCodeUtils.isGlucoseMeasurement(rec.getCode
()))
            .collect(Collectors.toList());
   }
}
```

```
package
com.diploma.linguistic_glucose_analyzer.service.filter.pro
vider;

import
com.diploma.linguistic_glucose_analyzer.service.filter.Rec
ordFilter;
import org.springframework.stereotype.Service;

import java.util.List;

public interface FiltersProvider {
   List<RecordFilter> getFilters();
}
```

```
package
com.diploma.linguistic_glucose_analyzer.service.filter.pro
vider;

import
com.diploma.linguistic_glucose_analyzer.constants.Linguis
ticChainConstants;
import
com.diploma.linguistic_glucose_analyzer.service.filter.Glu
coseOnlyRecordFilter;
import
com.diploma.linguistic_glucose_analyzer.service.filter.Min
MeasuresPerDayRecordFilter;
import
com.diploma.linguistic_glucose_analyzer.service.filter.Rec
ordFilter;
import org.springframework.stereotype.Service;

import java.util.ArrayList;
import java.util.List;

@Service
public class GlucoseFiltersProvider implements
FiltersProvider {
   @Override
   public List<RecordFilter> getFilters() {
      List<RecordFilter> filters = new ArrayList<>();

      filters.add(new GlucoseOnlyRecordFilter());
      filters.add(new
MinMeasuresPerDayRecordFilter(LinguisticChainConstan
ts.MIN_MEASURES_PER_DAY_PER_PERSON));

      return filters;
   }
}
```

```
package
com.diploma.linguistic_glucose_analyzer.service.matrix;

import
com.diploma.linguistic_glucose_analyzer.model.GlucoseD
ataRecord;
import
com.diploma.linguistic_glucose_analyzer.service.matrix.m
```

```
odel.PredictionMatrix;
import org.springframework.stereotype.Service;

import java.util.List;

public interface PredictionMatrixFactory {
   PredictionMatrix getMatrix(List<GlucoseDataRecord>
records, int chainLength);
}
```

```
package
com.diploma.linguistic_glucose_analyzer.service.matrix.m
odel;

import lombok.AllArgsConstructor;
import lombok.Data;

import java.util.HashMap;
import java.util.Map;

@Data
@AllArgsConstructor
public class PredictionMatrix {
   private int chainLength;
   private Map<String, Map<Character, Double>>
appearanceChances;
   private Map<String, Integer> chainOccasions;
   private Map<String, Map<Character, Integer>>
charAfterChainOccasions;

   public double getAppearanceChance(String chain, char
expectedSymbol) {
      if (chain == null || appearanceChances == null ||
chain.length() != chainLength) {
         throw new IllegalArgumentException();
      }

      return
appearanceChances.get(chain).get(expectedSymbol);
   }

   public Map<String, Double>
getPossibleChainsBeforeSymbol(char symbol) {
      Map<String, Integer> possibleChains = new
HashMap<>();

      for (Map.Entry<String, Map<Character, Integer>>
occasions : charAfterChainOccasions.entrySet()) {
         String chain = occasions.getKey();
         Integer charOccasions =
occasions.getValue().get(symbol);

         if (charOccasions != null) {
            possibleChains.put(chain, charOccasions);
         }
      }

      int overallOccasions = possibleChains
            .values()
            .stream()
            .mapToInt(x -> x)
            .sum();

      Map<String, Double> possibleChainsChances = new
HashMap<>();

      for (Map.Entry<String, Integer> possibleChain :
possibleChains.entrySet()) {
         possibleChainsChances.put(possibleChain.getKey(),
(double) possibleChain.getValue() / overallOccasions);
      }

      return possibleChainsChances;
   }
}
```

```
package
com.diploma.linguistic_glucose_analyzer.service.matrix.i
mpl;

import
com.diploma.linguistic_glucose_analyzer.model.GlucoseD
ataRecord;
import
com.diploma.linguistic_glucose_analyzer.service.Linguisti
cChainService;
import
com.diploma.linguistic_glucose_analyzer.service.matrix.Pr
edictionMatrixFactory;
import
com.diploma.linguistic_glucose_analyzer.service.matrix.m
odel.PredictionMatrix;
import lombok.extern.slf4j.Slf4j;
import
org.springframework.beans.factory.annotation.Autowired
;
import org.springframework.stereotype.Service;

import java.util.*;
import java.util.stream.Collectors;

@Slf4j
@Service
public class PredictionMatrixFactoryImpl implements
PredictionMatrixFactory {

   @Autowired
   private LinguisticChainService linguisticChainService;

   @Override
   public PredictionMatrix
getMatrix(List<GlucoseDataRecord> records, int
chainLength) {
      Collection<List<GlucoseDataRecord>>
filteredRecords = getRecordsByPerson(records);
      List<String> personChains =
getChainsByPerson(filteredRecords);

      Map<String, Integer> chainOccasions = new
HashMap<>();
      Map<String, Map<Character, Integer>>
charAfterChainOccasions = new HashMap<>();

      for (String personChain : personChains) {
         for (int i = chainLength; i < personChain.length();
i++) {
```

```
            String chain = personChain.substring(i - chainLength, i);
            Integer numSeen = chainOccasions.get(chain);

            if (numSeen == null) {
                numSeen = 0;
            }

            chainOccasions.put(chain, ++numSeen);

            Map<Character, Integer> numCharsSeen = charAfterChainOccasions.get(chain);

            if (numCharsSeen == null) {
                numCharsSeen = new HashMap<>();
            }

            char currSymbol = personChain.charAt(i);
            Integer numSymbolSeen = numCharsSeen.get(currSymbol);

            if (numSymbolSeen == null) {
                numSymbolSeen = 0;
            }

            numCharsSeen.put(currSymbol, ++numSymbolSeen);
            charAfterChainOccasions.put(chain, numCharsSeen);
        }
    }

    Map<String, Map<Character, Double>> appearanceChances = new HashMap<>();

    for (Map.Entry<String, Integer> occasions : chainOccasions.entrySet()) {
        Map<Character, Double> currChainChances = new HashMap<>();

        String chain = occasions.getKey();
        int overallChainOccasions = occasions.getValue();

        Map<Character, Integer> charsOccasions = charAfterChainOccasions.get(chain);

        for (Map.Entry<Character, Integer> charOccasions : charsOccasions.entrySet()) {
            currChainChances.put(charOccasions.getKey(), (double) charOccasions.getValue() / overallChainOccasions);
        }

        appearanceChances.put(chain, currChainChances);
    }

    return new PredictionMatrix(chainLength, appearanceChances, chainOccasions, charAfterChainOccasions);
  }

  private Collection<List<GlucoseDataRecord>> getRecordsByPerson(List<GlucoseDataRecord> records) {
    return records.stream()
        .filter(record -> record.getPerson().getId() != 0L)
        .collect(Collectors.groupingBy(GlucoseDataRecord::getPerson))
        .values();
  }

  private List<String> getChainsByPerson(Collection<List<GlucoseDataRecord>> filteredRecords) {
    return filteredRecords.stream()
        .map(personRecords -> linguisticChainService.getChain(personRecords))
        .collect(Collectors.toList());
  }
}
```

```
package com.diploma.linguistic_glucose_analyzer.auth;

import org.springframework.security.core.annotation.AuthenticationPrincipal;

import java.lang.annotation.*;

@Target({ElementType.PARAMETER, ElementType.TYPE})
@Retention(RetentionPolicy.RUNTIME)
@Documented
@AuthenticationPrincipal
public @interface CurrentUser {
  //Can be used in controllers to access the currently authenticated user
}
```

```
package com.diploma.linguistic_glucose_analyzer.auth;

import com.diploma.linguistic_glucose_analyzer.dao.UserDAO;
import com.diploma.linguistic_glucose_analyzer.model.User;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

@Service
public class CustomUserDetailsService implements UserDetailsService {

  @Autowired
  private UserDAO userDAO;
```

```
    @Override
    @Transactional
    public UserDetails loadUserByUsername(String
usernameOrEmail)
            throws UsernameNotFoundException {
        // Let people login with either username or email
        User user =
userDAO.getUserByUsernameOrEmail(usernameOrEmail
)
                .orElseThrow(() ->
                        new UsernameNotFoundException("User not
found with username or email : " + usernameOrEmail)
                );

        return UserPrincipal.create(user);
    }

    // This method is used by JWTAuthenticationFilter
    @Transactional
    public UserDetails loadUserById(Long id) {
        User user = userDAO.findById(id).orElseThrow(
                () -> new UsernameNotFoundException("User
not found with id : " + id)
        );

        return UserPrincipal.create(user);
    }
}
```

```
package com.diploma.linguistic_glucose_analyzer.auth;

import lombok.extern.slf4j.Slf4j;
import
org.springframework.security.core.AuthenticationExcepti
on;
import
org.springframework.security.web.AuthenticationEntryP
oint;
import org.springframework.stereotype.Component;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@Slf4j
@Component
public class JwtAuthenticationEntryPoint implements
AuthenticationEntryPoint {

    @Override
    public void commence(HttpServletRequest
httpServletRequest,
                         HttpServletResponse httpServletResponse,
                         AuthenticationException e) throws
IOException {
        log.error("Responding with unauthorized error.
Message - {}", e.getMessage());

httpServletResponse.sendError(HttpServletResponse.SC_
UNAUTHORIZED, e.getMessage());
    }
}
```

```
package com.diploma.linguistic_glucose_analyzer.auth;

import lombok.extern.slf4j.Slf4j;
import
org.springframework.beans.factory.annotation.Autowired
;
import
org.springframework.security.authentication.UsernameP
asswordAuthenticationToken;
import
org.springframework.security.core.context.SecurityConte
xtHolder;
import
org.springframework.security.core.userdetails.UserDetail
s;
import
org.springframework.security.web.authentication.WebAu
thenticationDetailsSource;
import org.springframework.util.StringUtils;
import
org.springframework.web.filter.OncePerRequestFilter;

import javax.servlet.FilterChain;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@Slf4j
public class JwtAuthenticationFilter extends
OncePerRequestFilter {

    @Autowired
    private JwtTokenProvider tokenProvider;

    @Autowired
    private CustomUserDetailsService
customUserDetailsService;

    protected void doFilterInternal(HttpServletRequest
request, HttpServletResponse response, FilterChain
filterChain) throws ServletException, IOException {
        try {
            String jwt = this.getJwtFromRequest(request);
            if (StringUtils.hasText(jwt) &&
this.tokenProvider.validateToken(jwt)) {
                Long userId =
this.tokenProvider.getUserIdFromJWT(jwt);
                UserDetails userDetails =
this.customUserDetailsService.loadUserById(userId);
                UsernamePasswordAuthenticationToken
authentication = new
UsernamePasswordAuthenticationToken(userDetails,
null, userDetails.getAuthorities());
                authentication.setDetails((new
WebAuthenticationDetailsSource()).buildDetails(request)
);

SecurityContextHolder.getContext().setAuthentication(au
thentication);
            }
        } catch (Exception e) {
```

```
        log.error("Could not set user authentication in
security context", e);
      }

      filterChain.doFilter(request, response);
   }

   private String getJwtFromRequest(HttpServletRequest
request) {
      String bearerToken =
request.getHeader("Authorization");
      if (StringUtils.hasText(bearerToken) &&
bearerToken.startsWith("Bearer ")) {
         return bearerToken.substring(7,
bearerToken.length());
      } else {
         return null;
      }
   }
}
```

```
package com.diploma.linguistic_glucose_analyzer.auth;

import io.jsonwebtoken.*;
import lombok.extern.slf4j.Slf4j;
import
org.springframework.beans.factory.annotation.Value;
import
org.springframework.security.core.Authentication;
import org.springframework.stereotype.Component;

import java.util.Date;

@Slf4j
@Component
public class JwtTokenProvider {

   @Value("${app.jwtSecret:JWTSuperSecretKey}")
   private String jwtSecret;

   @Value("${app.jwtExpirationInMs:3600000}")
   private int jwtExpirationInMs;

   public String generateToken(Authentication
authentication) {

      UserPrincipal userPrincipal = (UserPrincipal)
authentication.getPrincipal();

      Date now = new Date();
      Date expiryDate = new Date(now.getTime() +
jwtExpirationInMs);

      return Jwts.builder()
            .setSubject(Long.toString(userPrincipal.getId()))
            .setIssuedAt(new Date())
            .claim("username", userPrincipal.getUsername())
            .setExpiration(expiryDate)
            .signWith(SignatureAlgorithm.HS512, jwtSecret)
            .compact();
   }

   public Long getUserIdFromJWT(String token) {
      Claims claims = Jwts.parser()
            .setSigningKey(jwtSecret)
            .parseClaimsJws(token)
            .getBody();

      return Long.parseLong(claims.getSubject());
   }

   public boolean validateToken(String authToken) {
      try {

Jwts.parser().setSigningKey(jwtSecret).parseClaimsJws(a
uthToken);
         return true;
      } catch (SignatureException e) {
         log.error("Invalid JWT signature");
      } catch (MalformedJwtException e) {
         log.error("Invalid JWT token");
      } catch (ExpiredJwtException e) {
         log.error("Expired JWT token");
      } catch (UnsupportedJwtException e) {
         log.error("Unsupported JWT token");
      } catch (IllegalArgumentException e) {
         log.error("JWT claims string is empty.");
      }
      return false;
   }
}
```

```
package com.diploma.linguistic_glucose_analyzer.auth;

import
com.diploma.linguistic_glucose_analyzer.model.User;
import com.fasterxml.jackson.annotation.JsonIgnore;
import lombok.AllArgsConstructor;
import lombok.EqualsAndHashCode;
import lombok.Getter;
import
org.springframework.security.core.GrantedAuthority;
import
org.springframework.security.core.authority.SimpleGrant
edAuthority;
import
org.springframework.security.core.userdetails.UserDetail
s;

import java.util.Arrays;
import java.util.Collection;
import java.util.List;

import static
com.diploma.linguistic_glucose_analyzer.constants.Lingui
sticChainConstants.DEFAULT_AUTHORITY;

@Getter
@EqualsAndHashCode
@AllArgsConstructor
public class UserPrincipal implements UserDetails {

   private Long id;
   private String firstName;
   private String lastName;
   private String username;
```

```
    @JsonIgnore
    private String email;
    @JsonIgnore
    private String password;
    private Collection<? extends GrantedAuthority>
authorities;

    public static UserPrincipal create(User user) {
        List<GrantedAuthority> authorities =
Arrays.asList(new
SimpleGrantedAuthority(DEFAULT_AUTHORITY));
        return new UserPrincipal(user.getId(),
user.getPerson().getName(),
user.getPerson().getSurname(), user.getLogin(),
user.getEmail(), user.getPassword(), authorities);
    }

    @Override
    public boolean isAccountNonExpired() {
        return true;
    }

    @Override
    public boolean isAccountNonLocked() {
        return true;
    }

    @Override
    public boolean isCredentialsNonExpired() {
        return true;
    }

    @Override
    public boolean isEnabled() {
        return true;
    }
}
```

```
package
com.diploma.linguistic_glucose_analyzer.controller;

import
com.diploma.linguistic_glucose_analyzer.auth.JwtTokenPr
ovider;
import
com.diploma.linguistic_glucose_analyzer.dto.request.Logi
nRequest;
import
com.diploma.linguistic_glucose_analyzer.dto.request.Sign
UpRequest;
import
com.diploma.linguistic_glucose_analyzer.dto.response.Jwt
AuthenticationResponse;
import
com.diploma.linguistic_glucose_analyzer.service.AuthServ
ice;
import
org.springframework.beans.factory.annotation.Autowired
;
import org.springframework.http.ResponseEntity;
import
org.springframework.security.authentication.Authenticati
onManager;
import
org.springframework.security.crypto.password.Password
Encoder;
import org.springframework.web.bind.annotation.*;

import javax.validation.Valid;

@RestController
@RequestMapping("/api/auth")
@CrossOrigin("http://localhost:4200")
public class AuthController {

    private AuthenticationManager authenticationManager;
    private PasswordEncoder passwordEncoder;
    private JwtTokenProvider tokenProvider;
    private AuthService authService;

    @Autowired
    public AuthController(AuthenticationManager
authenticationManager, PasswordEncoder
passwordEncoder, JwtTokenProvider tokenProvider,
AuthService authService) {
        this.authenticationManager = authenticationManager;
        this.passwordEncoder = passwordEncoder;
        this.tokenProvider = tokenProvider;
        this.authService = authService;
    }

    @PostMapping("/signin")
    public ResponseEntity<?> authenticateUser(@Valid
@RequestBody LoginRequest loginRequest) {
        return ResponseEntity.ok(new
JwtAuthenticationResponse(authService.authenticateUse
r(
                loginRequest, authenticationManager,
tokenProvider)));
    }

    @PostMapping("/signup")
    public ResponseEntity<?> registerUser(@Valid
@RequestBody SignUpRequest signUpRequest) {
        return authService.registerUser(signUpRequest,
passwordEncoder);
    }
}
```

```
package
com.diploma.linguistic_glucose_analyzer.controller;

import
com.diploma.linguistic_glucose_analyzer.auth.CurrentUse
r;
import
com.diploma.linguistic_glucose_analyzer.auth.UserPrincip
al;
import
com.diploma.linguistic_glucose_analyzer.model.GlucoseD
ataRecord;
import
com.diploma.linguistic_glucose_analyzer.model.Person;
import
com.diploma.linguistic_glucose_analyzer.model.User;
import
com.diploma.linguistic_glucose_analyzer.service.GlucoseF
ileService;
import
com.diploma.linguistic_glucose_analyzer.service.GlucoseS
```

```
ervice;
import
com.diploma.linguistic_glucose_analyzer.service.UserServ
ice;
import lombok.extern.slf4j.Slf4j;
import
org.springframework.beans.factory.annotation.Autowired
;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import
org.springframework.security.core.context.SecurityConte
xtHolder;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.multipart.MultipartFile;

import javax.validation.Valid;
import java.io.File;
import java.util.List;

@RestController
@RequestMapping("/api/glucose")
@CrossOrigin("http://localhost:4200")
@Slf4j
public class GlucoseController {

  @Autowired
  private GlucoseService glucoseService;

  @Autowired
  private GlucoseFileService glucoseFileService;

  @Autowired
  private UserService userService;

  @GetMapping()
  public ResponseEntity<?> getAllGlucoseRecords() {
    return ResponseEntity.ok(glucoseService.getAll());
  }

  @GetMapping("/{id}")
  public ResponseEntity<?>
getGlucoseRecord(@PathVariable long id) {
    return
ResponseEntity.ok(glucoseService.getById(id));
  }

  @GetMapping("/person/{id}")
  public ResponseEntity<?>
getGlucoseRecordByPersonId(@PathVariable long id) {
    return
ResponseEntity.ok(glucoseService.getRecordsByPerson(i
d));
  }

  @GetMapping("/user")
  public ResponseEntity<?>
getGlucoseRecordByPersonId() {
    Object potentialPrincipal =
SecurityContextHolder.getContext().getAuthentication().g
etPrincipal();

    if (potentialPrincipal instanceof UserPrincipal) {
      return
ResponseEntity.ok(glucoseService.getRecordsByUser(((U
serPrincipal) potentialPrincipal).getId()));
    }
    return ResponseEntity.ok().build();
  }

  @PostMapping
  public ResponseEntity<?> createGlucoseRecord(@Valid
@RequestBody GlucoseDataRecord record) {
    Object potentialPrincipal =
SecurityContextHolder.getContext().getAuthentication().g
etPrincipal();

    if (potentialPrincipal instanceof UserPrincipal) {
      Person person =
userService.getById(((UserPrincipal)
potentialPrincipal).getId()).getPerson();
      record.setPerson(person);
    }

    return
ResponseEntity.ok(glucoseService.save(record));
  }

  @PostMapping("/fileUpload")
  public ResponseEntity<?>
uploadGlucoseRecordsFromFile(@RequestParam("file")
MultipartFile file) {
    log.debug("Got file: {}", file);

    try {
      if (!file.isEmpty()) {
        try {
          byte[] bytes = file.getBytes();
          String completeData = new String(bytes);
          String[] rows = completeData.split("\\n");
          List<GlucoseDataRecord> records =
glucoseFileService.getRecords(rows);

          Object potentialPrincipal =
SecurityContextHolder.getContext().getAuthentication().g
etPrincipal();

          if (potentialPrincipal instanceof UserPrincipal)
{
            Person person =
userService.getById(((UserPrincipal)
potentialPrincipal).getId()).getPerson();

            for (GlucoseDataRecord record: records) {
              record.setPerson(person);
            }
          }

          glucoseService.saveAll(records);
        } catch (Exception e) {
          throw new RuntimeException("FAIL!");
        }

        return
ResponseEntity.status(HttpStatus.OK).body("Successfully
uploaded!");
      }
    } catch (Exception e) {
      return
ResponseEntity.status(HttpStatus.EXPECTATION_FAILED)
.body("Failed to upload!");
    };
```

```
        return ResponseEntity.ok().build();
    }

    @PutMapping
    public ResponseEntity<?>
updateGlucoseRecord(@Valid @RequestBody
GlucoseDataRecord record) {
        Object potentialPrincipal =
SecurityContextHolder.getContext().getAuthentication().g
etPrincipal();

        if (potentialPrincipal instanceof UserPrincipal) {
            Person person =
userService.getById(((UserPrincipal)
potentialPrincipal).getId()).getPerson();
            record.setPerson(person);
        }

        glucoseService.save(record);
        return ResponseEntity.ok().build();
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<?>
deleteGlucoseRecord(@PathVariable long id) {
        glucoseService.deleteById(id);
        return ResponseEntity.ok().build();
    }
}
```

```
package
com.diploma.linguistic_glucose_analyzer.controller;

import
com.diploma.linguistic_glucose_analyzer.auth.UserPrincip
al;
import
com.diploma.linguistic_glucose_analyzer.service.Predictio
nService;
import lombok.extern.slf4j.Slf4j;
import
org.springframework.beans.factory.annotation.Autowired
;
import org.springframework.http.ResponseEntity;
import
org.springframework.security.core.context.SecurityConte
xtHolder;
import
org.springframework.web.bind.annotation.CrossOrigin;
import
org.springframework.web.bind.annotation.GetMapping;
import
org.springframework.web.bind.annotation.RequestMappi
ng;
import
org.springframework.web.bind.annotation.RestController
;

@RestController
@RequestMapping("/api/prediction")
@CrossOrigin("http://localhost:4200")
@Slf4j
public class PredictionController {

    @Autowired
    private PredictionService predictionService;

    @GetMapping()
    public ResponseEntity<?> getPossibleFutureGlucose() {
        Object potentialPrincipal =
SecurityContextHolder.getContext().getAuthentication().g
etPrincipal();

        if (potentialPrincipal instanceof UserPrincipal) {
            return
ResponseEntity.ok(predictionService.getPossibleFutureGl
ucoseValueForUser(((UserPrincipal)
potentialPrincipal).getId()));
        }

        return ResponseEntity.ok().build();
    }
}
```

```
package
com.diploma.linguistic_glucose_analyzer.controller;

import
com.diploma.linguistic_glucose_analyzer.auth.UserPrincip
al;
import
com.diploma.linguistic_glucose_analyzer.dto.UserSummar
yDTO;
import
com.diploma.linguistic_glucose_analyzer.service.UserServ
ice;
import
org.springframework.beans.factory.annotation.Autowired
;
import org.springframework.http.ResponseEntity;
import
org.springframework.security.core.context.SecurityConte
xtHolder;
import org.springframework.web.bind.annotation.*;

import javax.validation.Valid;

@RestController
@RequestMapping("/api/account/users")
@CrossOrigin("http://localhost:4200")
public class UserController {

    @Autowired
    private UserService userService;

    @GetMapping()
    public ResponseEntity<?> loadAllUsers() {
        return ResponseEntity.ok(userService.getAll());
    }

    @GetMapping("/{id}")
    public ResponseEntity<?> getUser(@PathVariable long
id) {
        return ResponseEntity.ok(userService.getById(id));
    }

    @GetMapping("/summary")
    public ResponseEntity<?> getCurrentUserSummary() {
```

```
        Object potentialPrincipal =
SecurityContextHolder.getContext().getAuthentication().g
etPrincipal();

        if (potentialPrincipal instanceof UserPrincipal) {
            return
ResponseEntity.ok(userService.getUserSummary(((UserP
rincipal) potentialPrincipal).getId()));
        }

        return ResponseEntity.ok().build();
    }

    @PutMapping("/summary")
    public ResponseEntity<?>
updateCurrentUserSummary(@Valid @RequestBody
UserSummaryDTO userSummary) {
        Object potentialPrincipal =
SecurityContextHolder.getContext().getAuthentication().g
etPrincipal();

        if (potentialPrincipal instanceof UserPrincipal) {
            userService.updateUserData(((UserPrincipal)
potentialPrincipal).getId(), userSummary);
            return ResponseEntity.ok().build();
        }

        return ResponseEntity.ok().build();
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<?> deleteUser(@PathVariable
long id) {
        userService.deleteById(id);
        return ResponseEntity.ok().build();
    }
}
```

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

“___” _____ 2020 р.

Веб-застосування для аналізу рівня глюкози у крові діабетиків лінгвістичним методом

Програма та методика тестування

КПІ.ІП-6126.045440.04.51

“ПОГОДЖЕНО”

Керівник проєкту:

_____ О.К. Очеретяний

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____І.О. Шаверський

Київ – 2020 року

ЗМІСТ

1 ОБ’ЄКТ ВИПРОБУВАНЬ

Веб-додаток для аналізу рівня глюкози лінгвістичним методом у крові діабетиків, який являє собою веб-сайт, створений на фреймворку Spring Boot з використанням фронт-енд фреймворку Angular 2+.

2 МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено наступне:

- функціонал, який підлягає тестуванню;
- функціонал, який не підлягає тестуванню;
- способи тестування;
- визначні критерії проходження тестів;
- процес тестування;
- вимоги середовища.

3 МЕТОДИ ТЕСТУВАННЯ

Тестування виконується методам White Box Testing та Black Box Testing. Перевіряється підлягає як і кодова база, так і сам програмний продукт на значення відповідності функціональним вимогам та не функціональним вимогам. Тестування проводилося на рівні «системного тестування», виключаючи інтеграційне та компонентне.

Використовуються наступні методи:

– функціональне тестування, у тому числі базове (на рівні Critical path test);

– тестування продуктивності програмного забезпечення, зокрема Stability testing (тестування стабільності) та Load testing (навантажувальне тестування);

– тестування інтерфейсу.

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконувалось засобами ручного тестування, засобу JMeter та популярної бібліотеки для тестування на мові програмування Java JUnit.

Працездатність web-ресурсу перевіряється шляхом:

– динамічного ручного тестування – введенням граничних та недопустимих значень в поля, які можна редагувати;

– динамічного ручного тестування на відповідність функціональним вимогам;

– статичного тестування коду;

– тестування web-ресурсу в різних web-браузерах;

– тестування при максимальному навантаженні;

– тестування стабільності роботи при різних умовах;

– тестування зручності використання;

– тестування інтерфейсу.

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

“___” _____ 2020 р.

Веб-застосування для аналізу рівня глюкози лінгвістичним методом у крові діабетиків

Керівництво користувача

КПІ.ІП-6126.045440.05.34

“ПОГОДЖЕНО”

Керівник проєкту:

_____ О.К. Очеретяний

Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

_____І.О. Шаверський

Київ – 2020 року

ЗМІСТ

1 ІНСТРУКЦІЯ КОРИСТУВАЧА

1.1 Верхня панель

Кожна сторінка цього веб-додатку має у своїй верхній частині спеціальну панель для навігації по сайту та виконання певних дій. Вигляд цієї панелі залежить від того чи авторизований користувач.

Вигляд панелі для неавторизованого користувача можна побачити на рисунку 1.1.

Рисунок 1.1 – Вигляд панелі для неавторизованого користувача

На даній панелі можна побачити 3 кнопки. Кнопка повернення на головний екран (Рис 1.2), при натисканні якої неавторизований користувач потрапляє на сторінку авторизації. Кнопка авторизації (Рис 1.3) підписана, як «Sign in», за допомогою якої користувач також опиняється на сторінці авторизації. Кнопка реєстрації (Рис 1.4) підписана, як «Sign up», яка при натисканні навігує користувача до сторінки реєстрації

Рисунок 1.2 – Кнопка повернення на головний екран

Рисунок 1.3 – Кнопка авторизації

Рисунок 1.4 – Кнопка реєстрації

Вигляд панелі для авторизованого користувача можна побачити на рисунку 1.5.

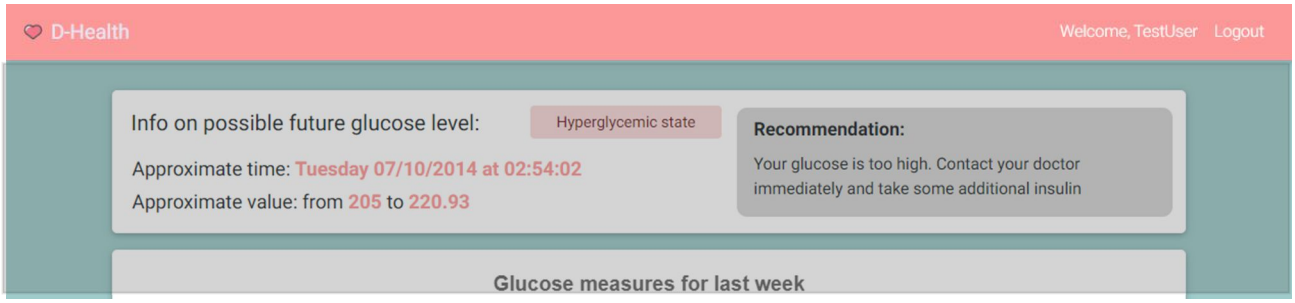


Рисунок 1.5 – Вигляд панелі для неавторизованого користувача

На даній панелі також 3 кнопки. Кнопка повернення на головний екран (Рис 1.2), при натисканні якої авторизований користувач потрапляє на головну сторінку. Кнопка профілю (Рис 1.6) підписана, як «Welcome, %ім'я користувача%», може перейти на сторінку користувача. Кнопка деавторизаццїї (Рис 1.7) підписана, як «Logout», яка при натисканні деавторизує користувача та повертає на сторінку авторизації.

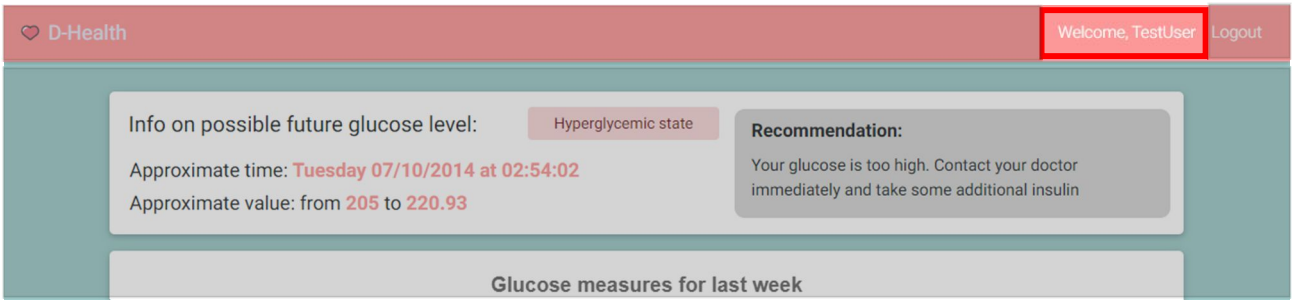


Рисунок 1.6 – Кнопка профілю

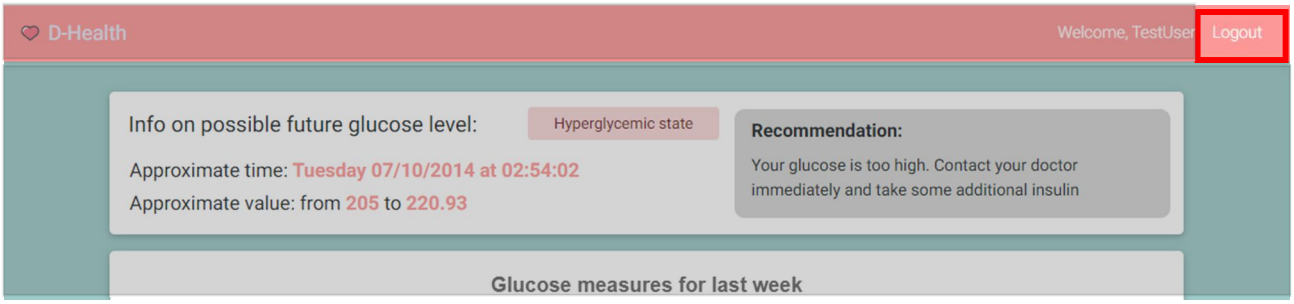


Рисунок 1.7 – Кнопка деавторизаццїї

1.2 Реєстрація та авторизація

Для доступу до веб-додатку користувачу спочатку потрібно зареєструватись. Реєстрація проходить на вікні реєстрації, вигляд якого можна побачити на рисунку 1.8. Для реєстрації користувач обов'язково має вказати унікальне ім'я користувача (не використане досі іншими користувача системи) у полі Username, наявний у користувача тип діабету (на вибір представлені варіанти Перший тип (First type), Другий тип (Second type), Діабет відсутній (No diabetes) у полі Diabetes Type, унікальну пошту користувача (не використану досі іншими користувача системи) у полі Email, свій пароль для доступу у систему у полі Password, що обмежений такими правилами:

- мінімум 8 символів;
- мінімум 1 літера верхнього регістру;
- мінімум 1 літера нижнього регістру;
- мінімум 1 цифра.

та повторно ввести цей пароль у полі Password Confirmation для його підтвердження. Також, за бажання, користувач може зазначити своє ім’я у полі First name, фамілію у полі Last name та дату народження у полі Birth Date. Після введення обов’язкових даних потрібно натиснути кнопку «Sign up» внизу вікна для підтвердження реєстрації. При невірно введених даних побачить поради біля полів або будь-якій іншій помилці користувач побачить повідомлення про дану помилку у поп-апі під надписом «Sign up» зверху сторінки.

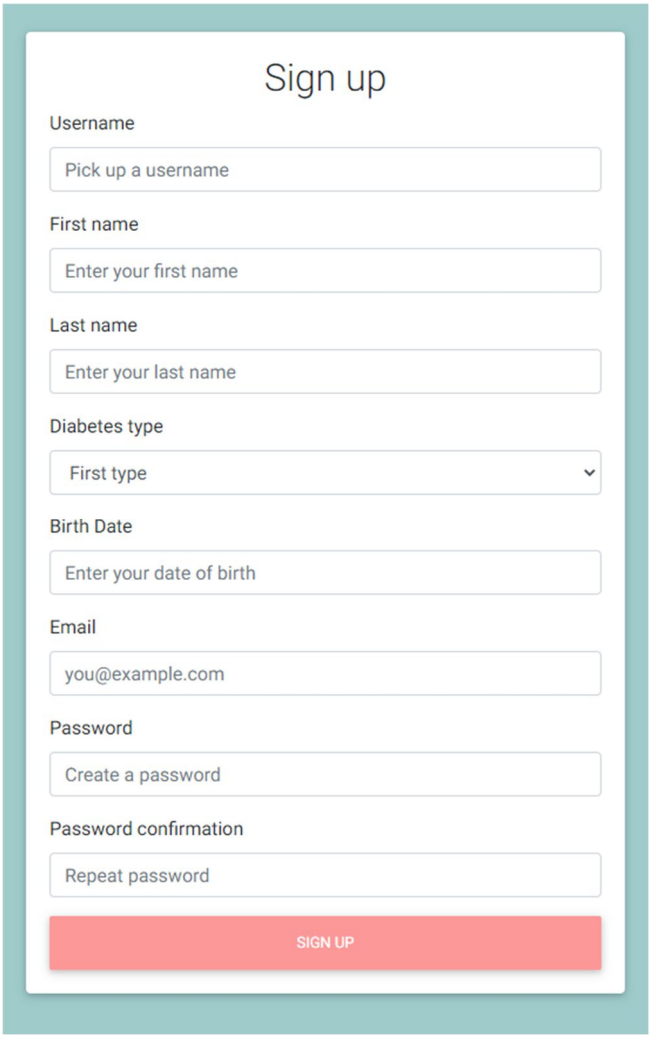


Рисунок 1.8 – Вікно реєстрації

Після успішної реєстрації користувач може авторизуватись у системі за допомогою вікна авторизації, вигляд якого можна побачити на рисунку 1.9. Для успішної авторизації користувача потрібно ввести у поле Username своє унікальне ім’я користувача, зареєстрованого на попередньому кроці та пароль

який ми вибирали там же у полі Password. Для підтвердження авторизації користувач має натиснути на кнопку Sign in внизу вікна. При невірно введених даних побачить поради біля полів або будь-якій іншій помилці користувач побачить повідомлення про дану помилку у поп-апі під надписом «Sign up» зверху сторінки. Після успішної авторизації користувач буде перенаправлений на головну сторінку.

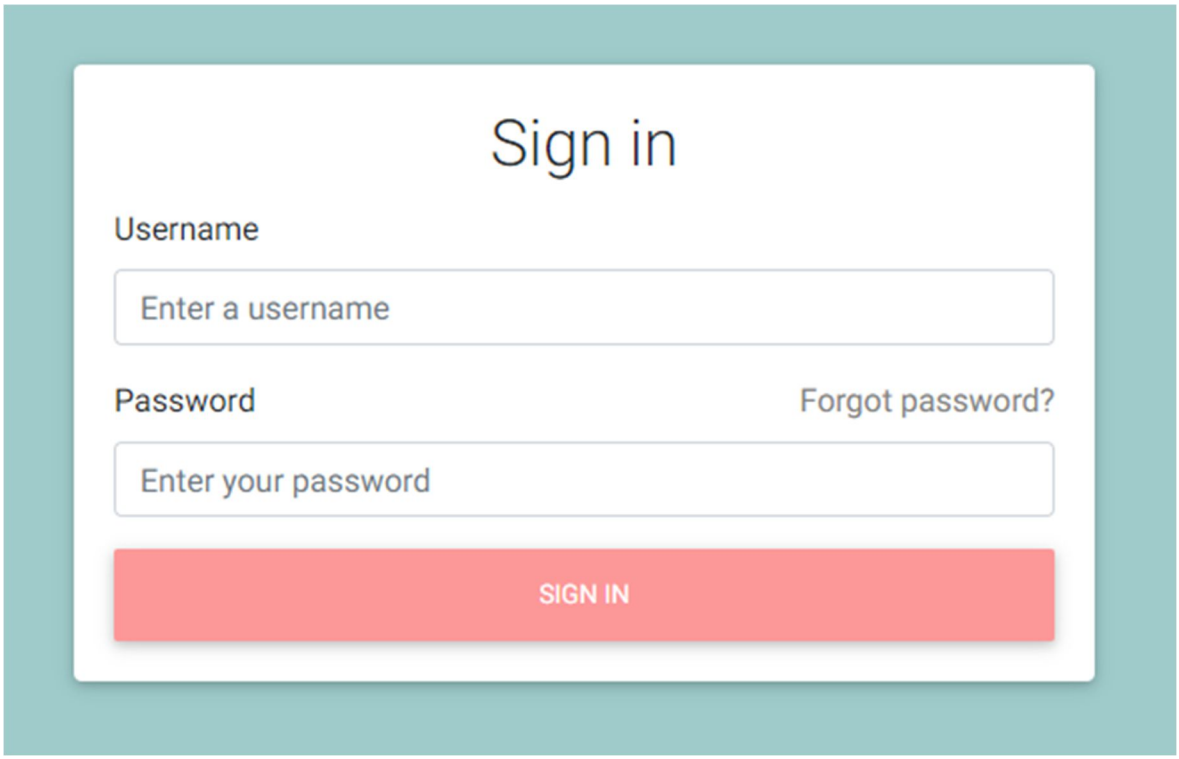


Рисунок 1.9 – Вікно авторизації

1.3 Головний екран

Головний екран містить у собі основний функціонал даного веб-застосунку. Вигляд даної сторінки можна побачити на рисунку 1.10. На даному екрані доступна таблиця значень глюкози з можливостями маніпулювання цими даними, графік значень глюкози за останню неділю та вікно передбачення можливого значення рівню глюкози через пів години після останнього значення у таблиці значень глюкози з рекомендаціями.

Рисунок 1.10 – Головний екран

Таблиця значень глюкози, вигляд якої можна побачити на рисунку 1.11, відображає записані користувачем, або якимось девайсом записи рівню глюкози. Таблиця має 4 колонки. Колонка з часом заміру, що підписана, як Time of measurement, колонка з значенням глюкози у форматі мг/дл, з назвою Value (mg/dl), колонка з інформаційним лейблом, який описує даний рівень глюкози та попереджує користувача про небезпечний рівень, що підписана, як Info та

технічна колонка без підпису у якій розміщені кнопки для маніпулювання даними.

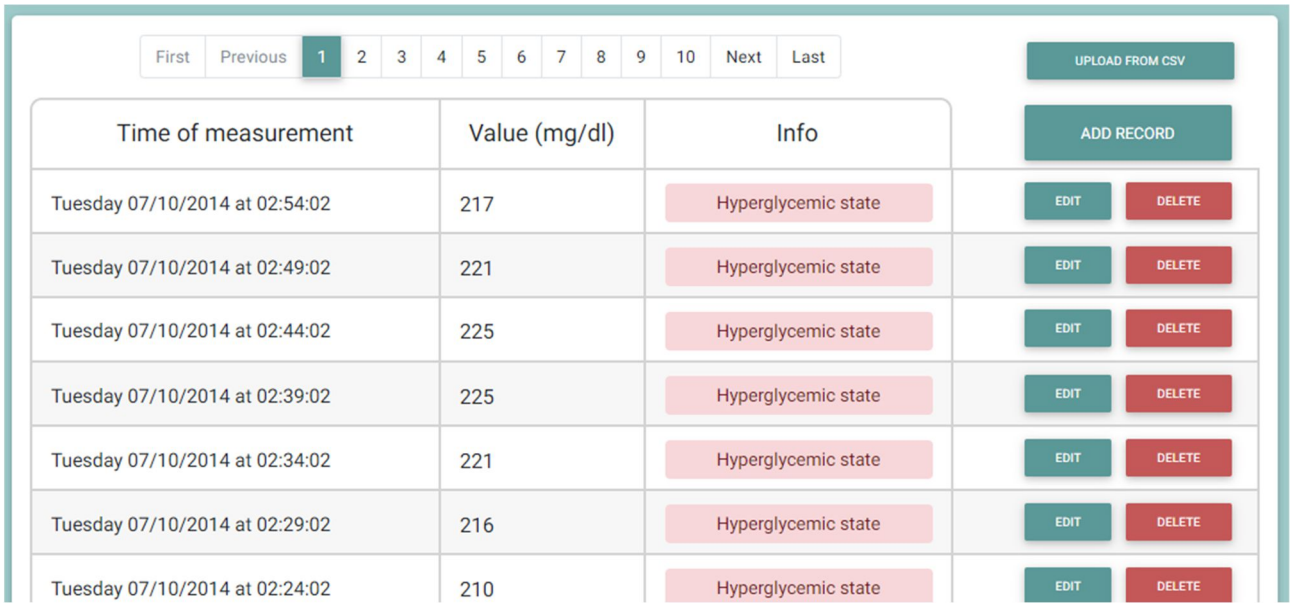


Рисунок 1.11 – Таблиця значень рівню глюкози

Записи у таблиці відсортовані за часом заміру. Таблиця оснащена пагінацією і перехід між цими сторінками виконується за допомогою панелі над таблицею, яку можна побачити на рисунку 1.12. На цій панелі можна перемикатись посторінково вибираючи певну сторінку, перемикатись на попередню та наступну сторінку за допомогою кнопок Previous та Next відповідно, а також переходити відразу на першу або останню сторінку таблиці використовуючи кнопки First та Last відповідно. Максимальна кількість записів на одній сторінці двадцять.

Рисунок 1.12 – Панель навігації сторінками таблиці

Додавання нових записів можливе двома способами. Перший це додавання за допомогою кнопки Add record, яку можна побачити на рисунку 1.13. При натисканні цієї кнопки до таблиці на першу позицію додається новий незаповнений запис (Рис 1.14) у якому ми змінюємо дані запису та маємо два варіанти подальших дій. Якщо, ми натиснемо кнопку Save, то запис буде збережено до бази даних, та ми вийдемо з режиму редагування. При натисканні кнопки Cancel цей незаповнений запис буде видалено, а введені дані не будуть збережені до бази даних.

Рисунок 1.13 – Кнопки Add record

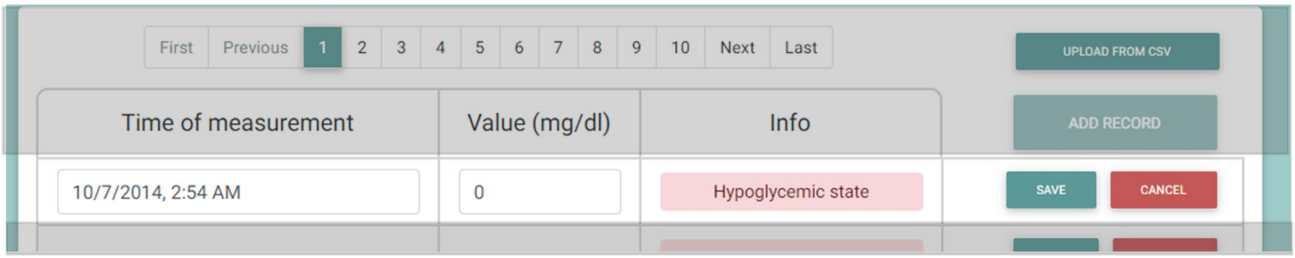


Рисунок 1.14– Новий незаповнений запис

Другий спосіб додавання нових засобів це кнопка Upload from CSV, яку можна побачити на рисунку 1.15, за допомогою якої можна завантажити засоби з коректного файлу формату .csv. Даний файл має бути з полями Date (Формат MM-DD-YYYY), Time (Формат hh:mm:ss), Glucose Value (У ммоль/л). Для завантаження потрібно просто натиснути на цю кнопку та вибрати файл підходящого формату. При успішному завантаженні значення будуть додані до таблиці.

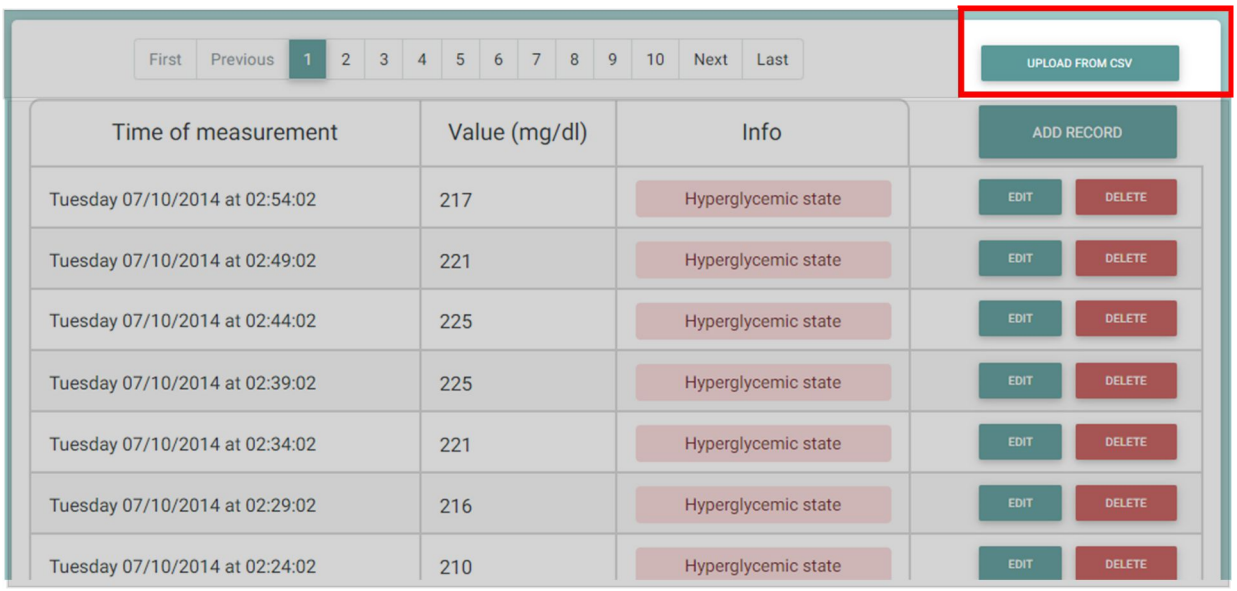


Рисунок 1.15 – Кнопка Upload from CSV

Також у нас є можливість маніпулювати уже наявними записами за допомогою кнопок Edit та Delete які можна побачити на рисунку 1.16. При натисканні на кнопку Edit поле зміниться і буде виглядати, як поле нового незаповненого запису (Рис. 1.14), тільки значення у полях будуть відповідати значенням відповідного запису. При натисканні Save зміни будуть збережені до

таблиці та бази. При натисканні Cancel зміни буде відмінено. При натисканні Delete буде виведено нове вікно підтвердження (Рис 1.17). У ньому запис буде видалено тільки при натисканні на кнопку Yes.

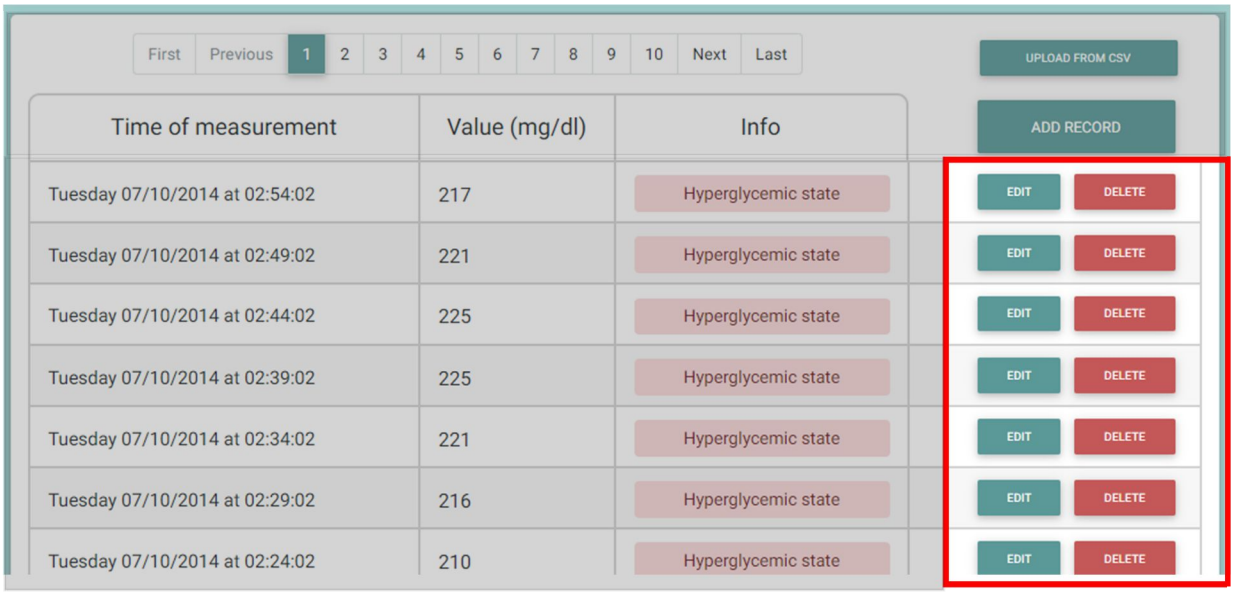


Рисунок 1.16 – Кнопки Edit та Delete

Рисунок 1.17 – Вікно підтвердження видалення запису

Графік записів рівню глюкози можна побачити на рисунку 1.18. Там відображені записи за останню неділю, де остання неділя рахується не від поточного часу, а від часу заміру останнього запису рівня глюкози.

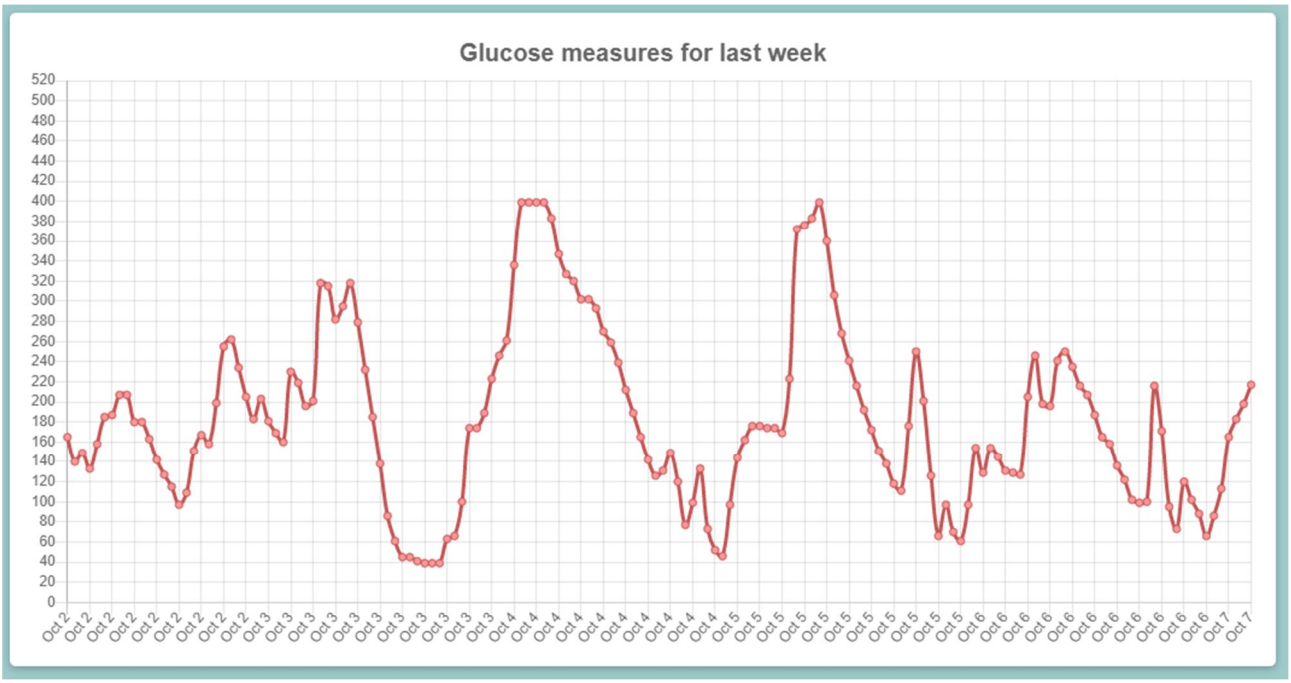


Рисунок 1.18 – Графік записів рівню глюкози

Зверху сторінки можна знайти поле передбачення можливого рівня глюкози, вигляд якого можна побачити на рисунку 1.19. На ньому представлена інформація про можливий рівень глюкози через пів години після останнього заміру рівня глюкози та рекомендація, щодо дій відповідних до цього можливого рівня глюкози. На верхній частині можна побачити лейбл відповідний до можливого значення, трохи нижче написаний орієнтовний час на який робиться передбачення, ще нижче можна побачити можливі рамки рівню глюкози. У правій частині видно рекомендацію, яка направляє на можливі дії, які в теорії можуть допомогти нормалізувати рівень глюкози. Ці дані оновлюються при завантаженні сторінки, або при маніпуляціях з записами з таблиці описаної раніше.

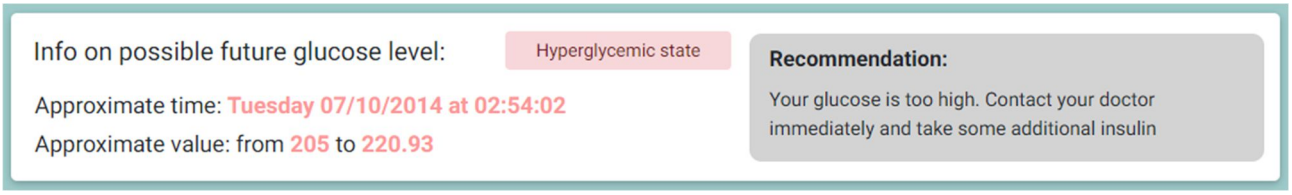


Рисунок 1.19– Поле передбачення можливого рівня глюкози

1.4 Екран користувача

Як, було зазначено вище, перейти на сторінку користувача можливо натиснувши на кнопку Welcome, %ім'я користувача%. Побачити вигляд цієї сторінки можна побачити на рисунку 1.20. На ній можна побачити усі ті ж дані, що можуть бути зазначені на сторінці реєстрації.

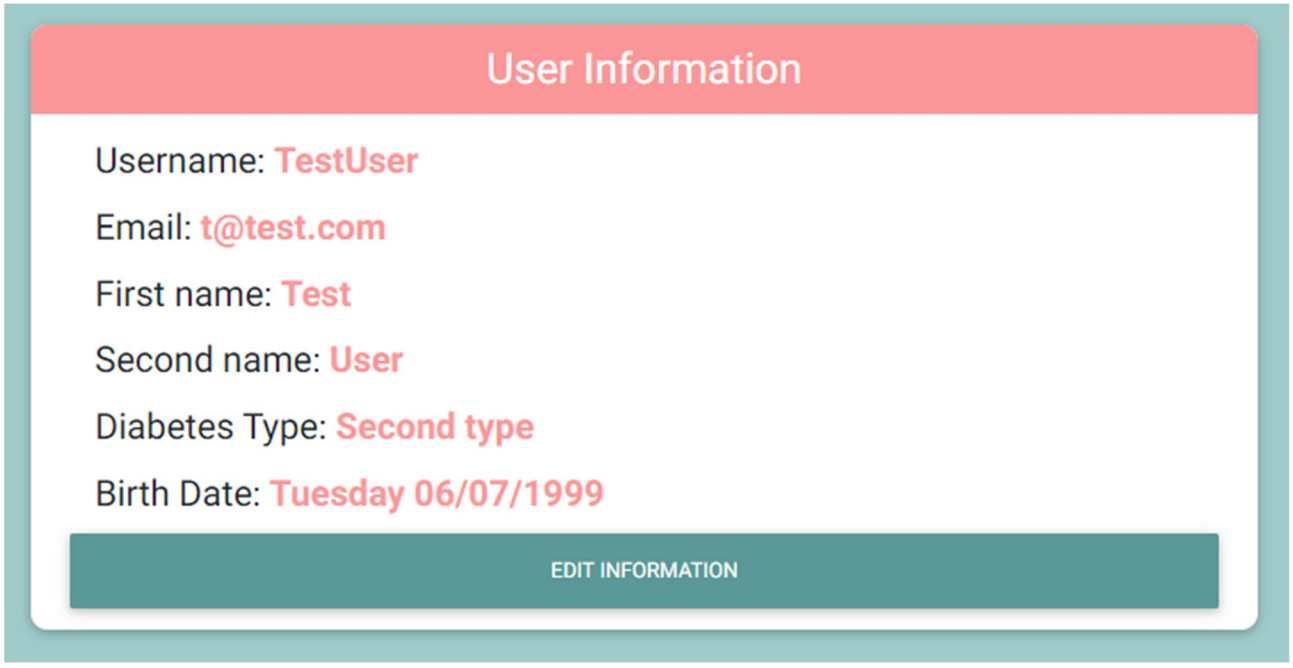


Рисунок 1.20– Профіль користувача

Ці дані можна редагувати, натиснувши на кнопку Edit Information, що розміщена внизу вікна профілю користувача. Вигляд поточного вікна зміниться і буде відповідати рисунку 1.21. Дані користувача будуть перенесені у відповідні поля для внесення інформації у даному вікні. Після редагування цих даних можна натиснути на кнопку Save. Після цього дані будуть збережені до бази і відображені на вікні профілю користувача. Також можна натиснути кнопку Cancel для відміни відповідних змін.

Рисунок 1.21– Редагування профілю користувача

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____ Олександр ПАВЛОВ

“___” _____ 2020 р.

Веб-застосування для аналізу рівня глюкози у крові діабетиків лінгвістичним методом

Графічний матеріал

КПІ.ІП-6126.045440.06.99

“ПОГОДЖЕНО”

Керівник проєкту:

_____ О.К. Очеретяний

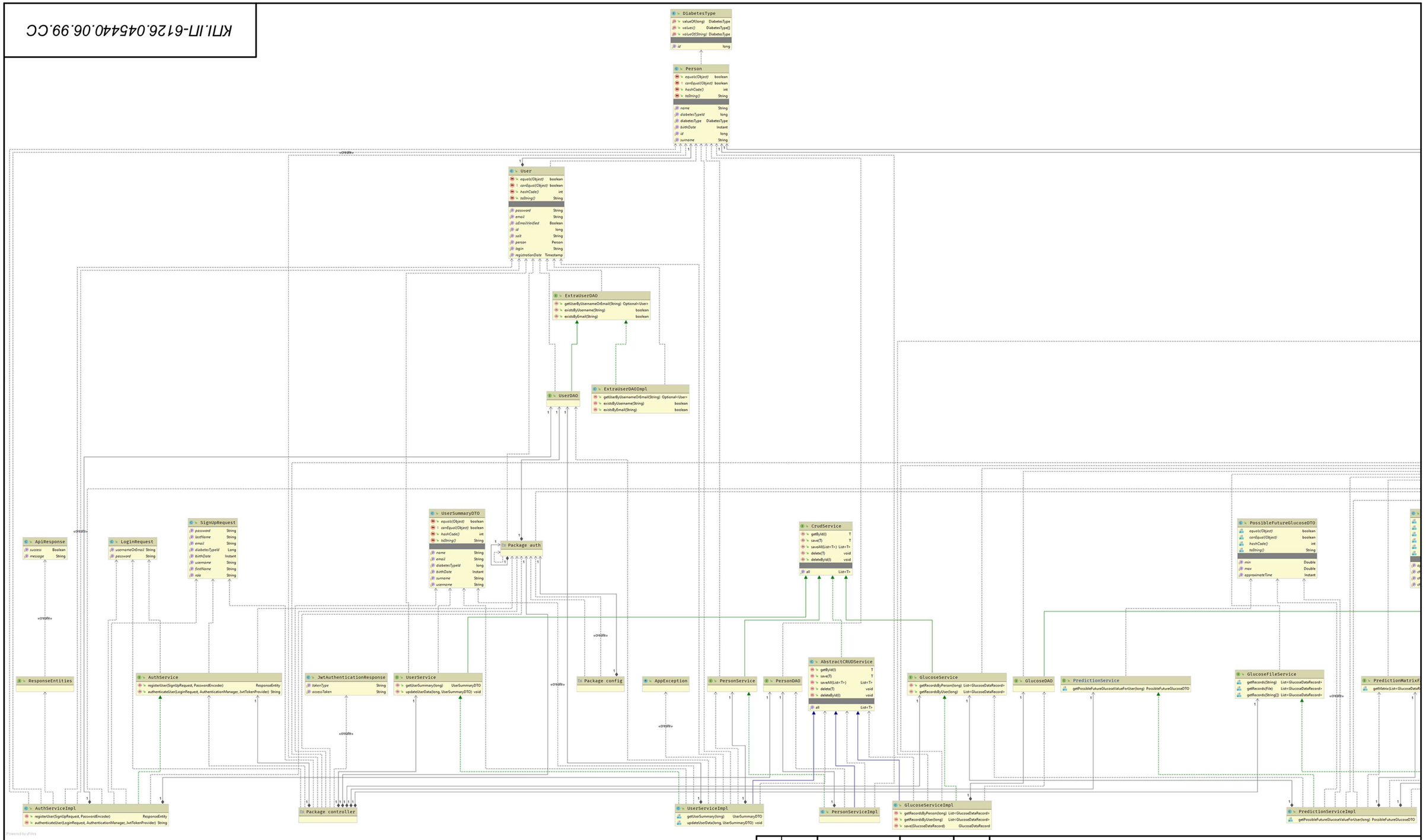
Нормоконтроль:

_____ К.І. Ліщук

Виконавець:

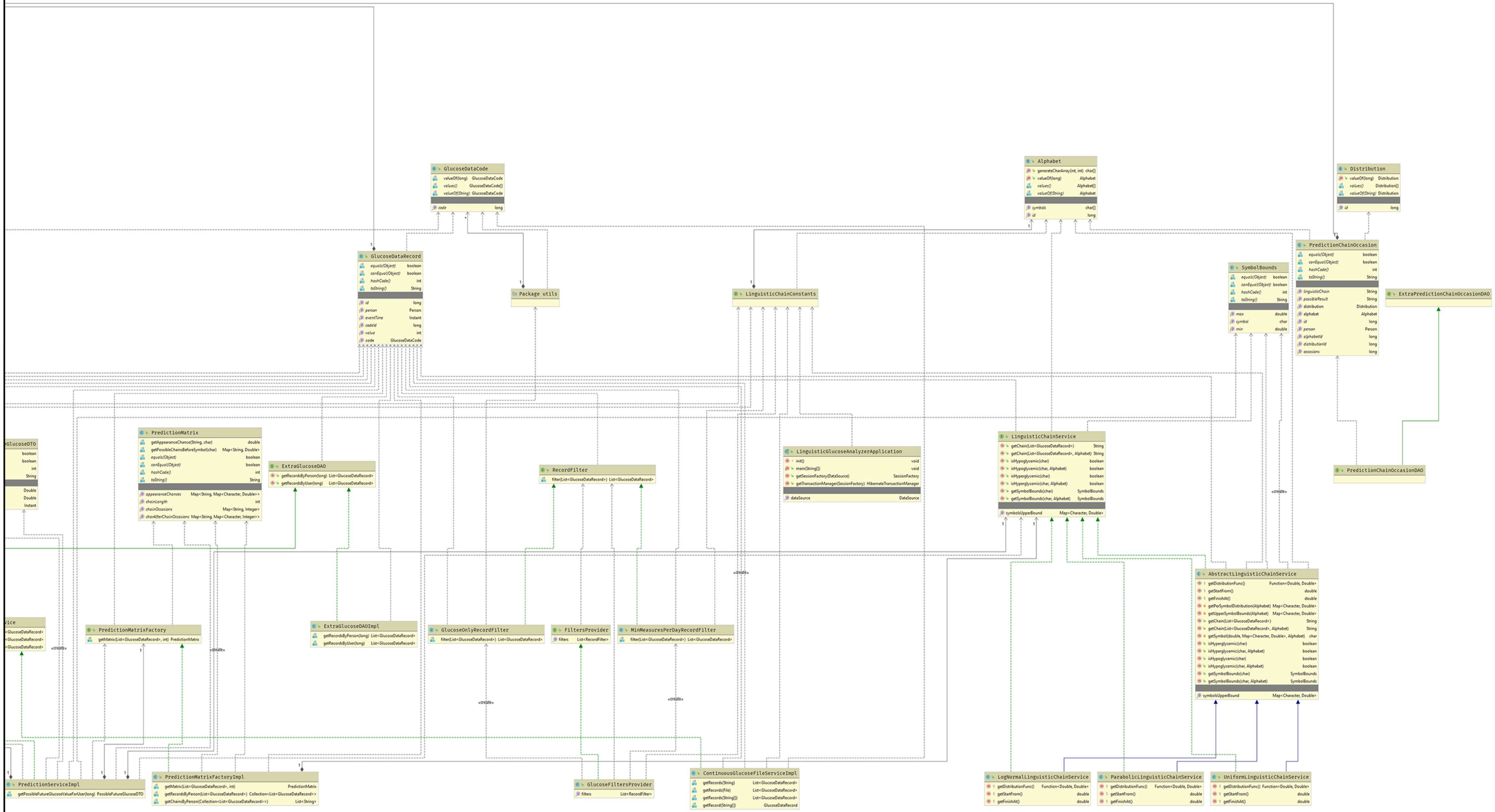
_____І.О. Шаверський

Київ – 2020 року

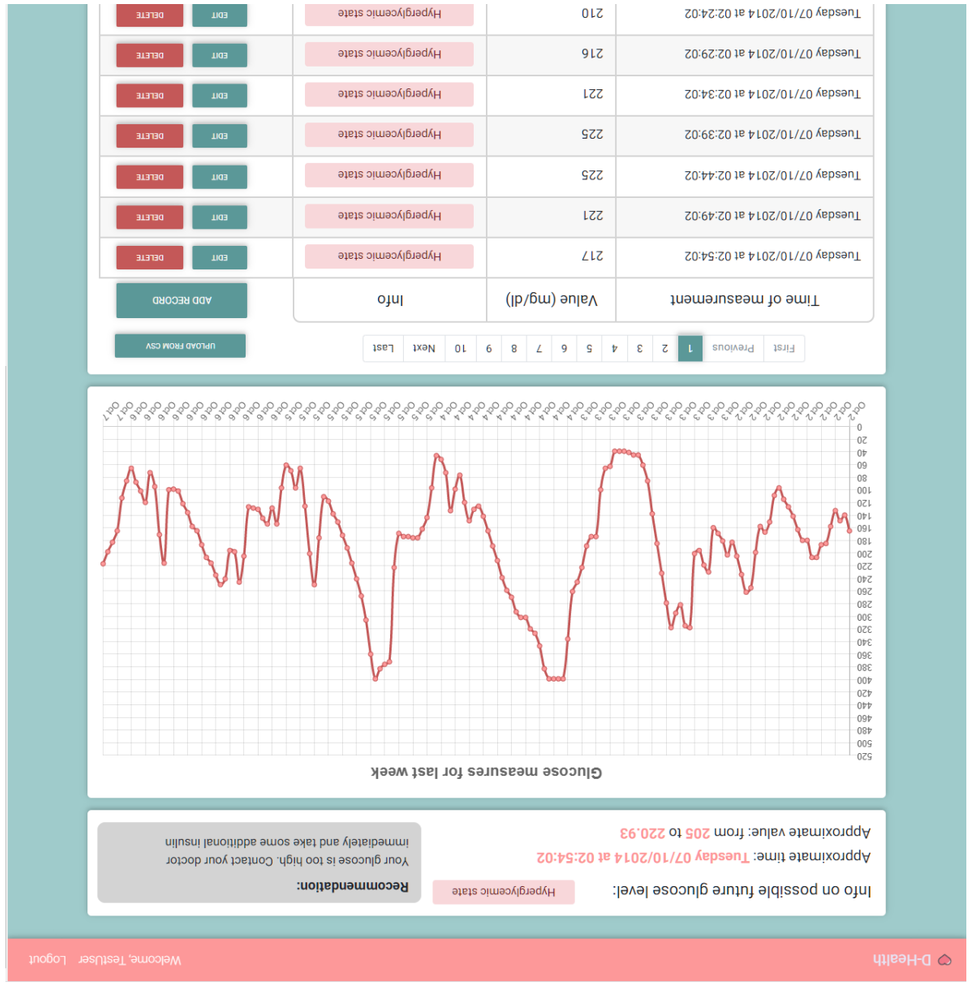


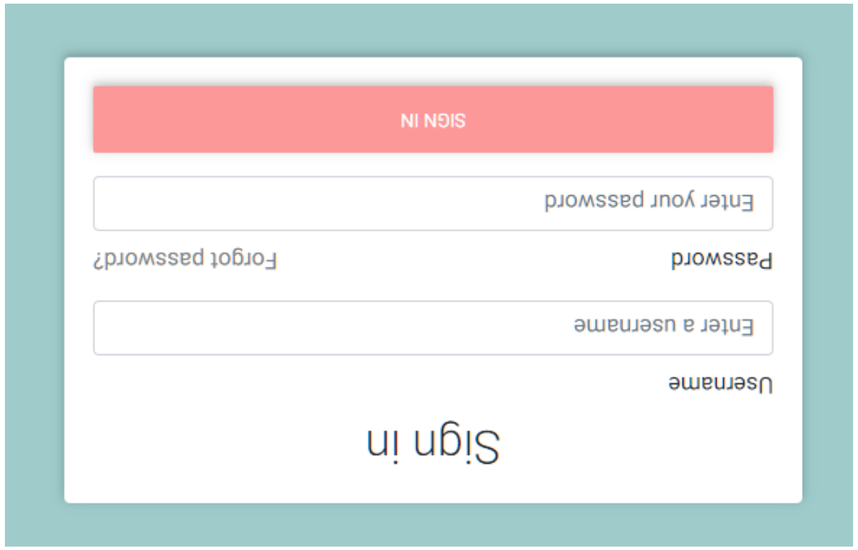
КПІ.ІП-6126.045440.06.99.СС

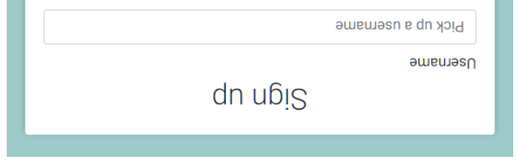
					КПІ.ІП-6126.045440.06.99.СС		
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна класів програмного забезпечення	Літера	Маса / Масштаб
Розробив		Шаверський І.О.					
Перевірив		Очеретяний О. К.				Аркуш 1	Аркушів 2
Т. кон.							
					Веб-застосування для аналізу рівня глюкози у крові діабетиків лінгвістичним методом	КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-61	
Н. кон.		Ліщук К.І.					
Затвердив		Очеретяний О. К.					



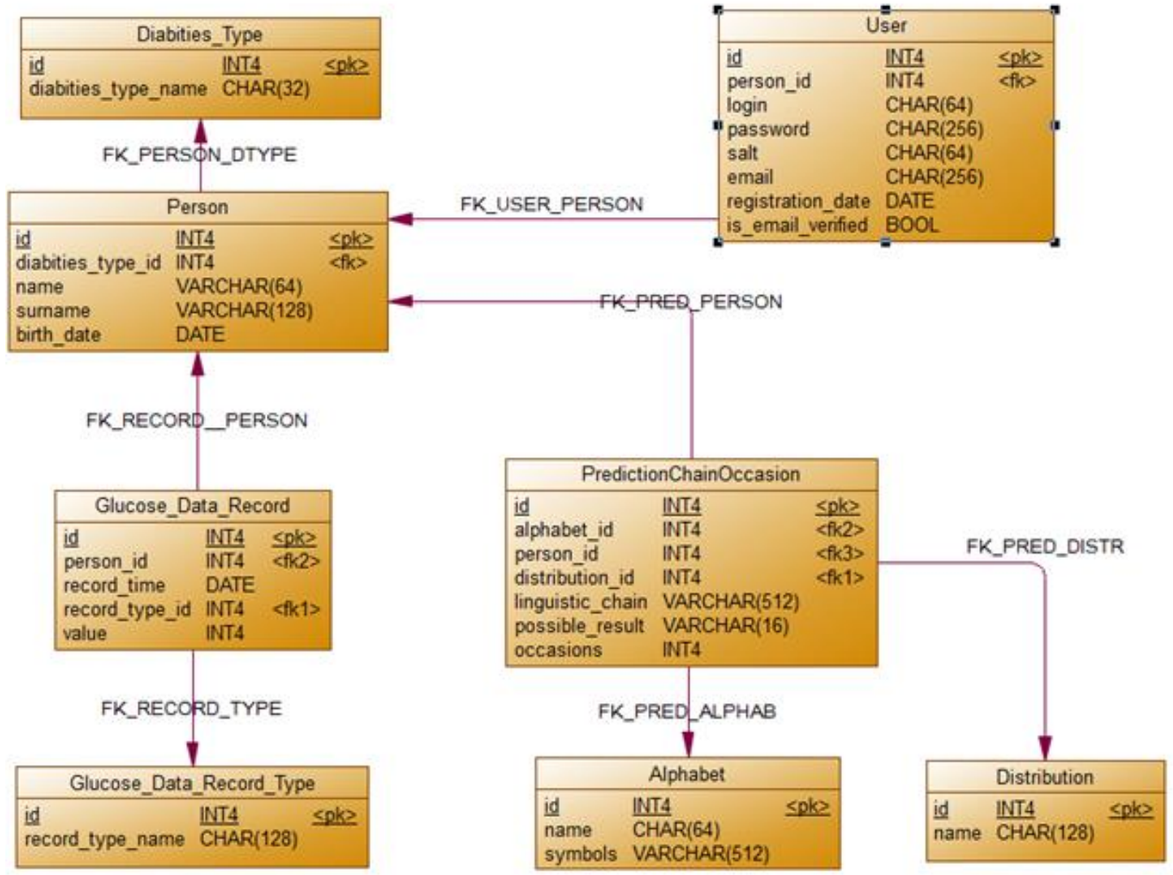
					КПІ.ІП-6126.045440.06.99.СС	
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна класів програмного забезпечення	Літера / Маса / Масштаб
Розробив		Шаверський І.О.				
Перевірив		Очеретяний О. К.				Аркуш 2 / Аркушів 2
Т. кон.						
Н. кон.		Ліщук К.І.			Веб-застосування для аналізу рівня глюкози у крові діабетиків лінгвістичним методом	КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-61
Затвердив		Очеретяний О. К.				



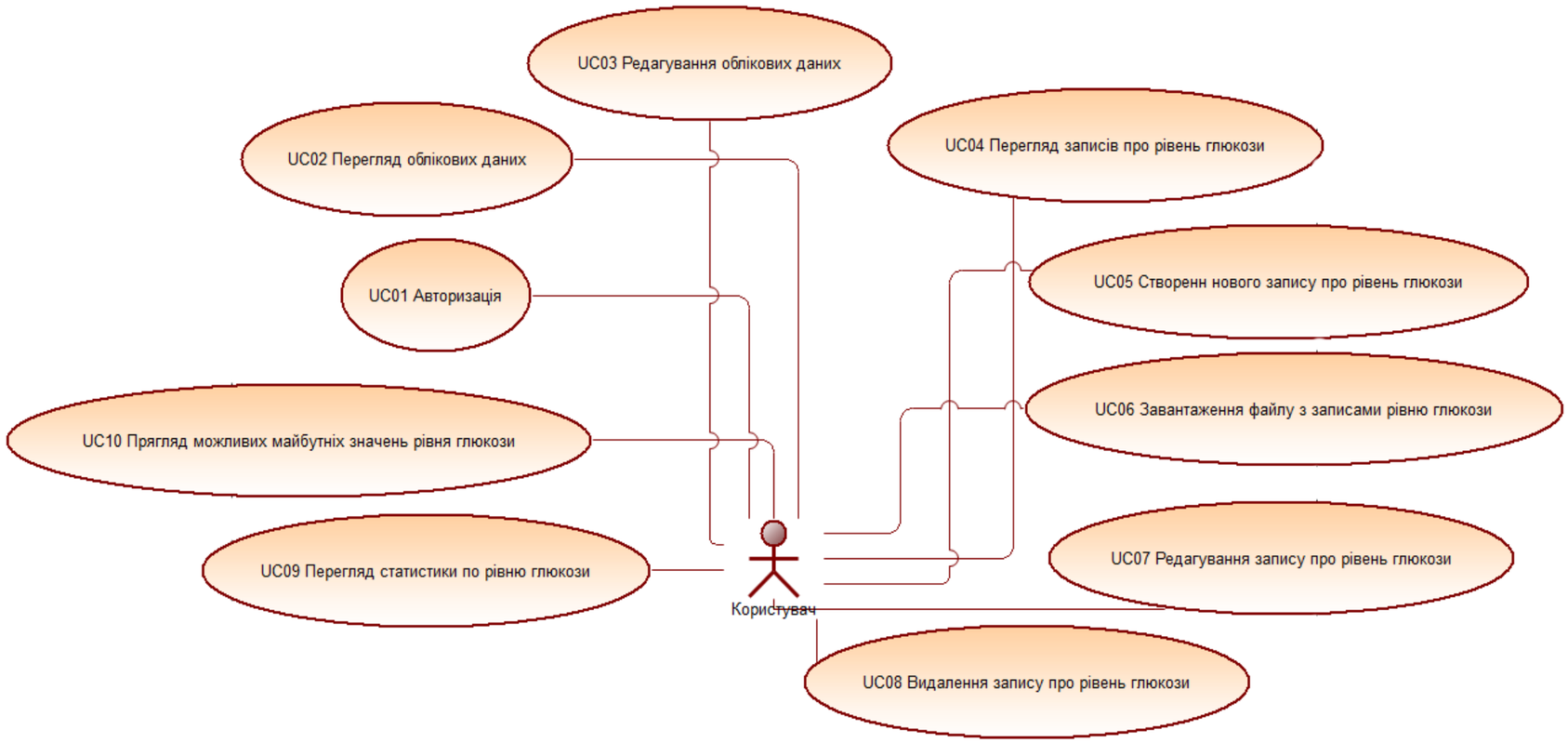




Зм.	Арк.	№ докумeнта	Підпис	Дата	КПІ.ІП-6126.045440.06.99.КЕ		
Розробив		Шаверський І.О.			Керування використанням екранних форм	Літера	Маса	Масштаб
Перевірив		Очеретний О. К.						
Т. контр.						Аркуш		Аркушів
Н. контр.		Лішук К.І.			Веб-застосування для аналізу рівня глюкози у крові діабетиків лінгвістичним методом	КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-61		
Затвердив		Очеретний О. К.						



					КПІ.ІП-6126.045440.06.99.СБД		
					Схема бази даних	Літера	Маса / Масштаб
Зм.	Арк.	№ документа	Підпис	Дата			
Розробив		Шаверський І.О.				Аркуш	Аркушів
Перевірив		Очеретяний О. К.			Веб-застосування для аналізу рівня глюкози у крові діабетиків лінгвістичним методом	КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-61	
Т. кон.							
Н. кон.		Ліщук К.І.					
Затвердив		Очеретяний О. К.					



					КПІ.ІП-6126.045440.06.99.СС	
					Схема структурна варіантів використання	Літера: — Маса: — Масштаб: —
Зм.	Арк.	№ документа	Підпис	Дата		Аркуш: — Аркушів: —
Розробив		Шаверський І.О.				
Перевірив		Очеретяний О. К.				
Т. кон.						
					Веб-застосування для аналізу рівня глюкози у крові діабетиків лінгвістичним методом	КПІ ім.Ігоря Сікорського Кафедра АСОІУ гр. ІП-61
Н. кон.		Ліщук К.І.				
Затвердив		Очеретяний О. К.				